AD NUMBER

ADB015343

LIMITATION CHANGES

TO:

Approved for public release; distribution is unlimited.

FROM:

Distribution authorized to U.S. Gov't. agencies only; Test and Evaluation; APR 1976. Other requests shall be referred to Air Force Armament Lab., Eglin AFB, FL.

AUTHORITY

ADTC ltr 3 Jul 1979

(18) (19) AFATL TR-76-45, BOOK-1

HELP - A MULTIMATERIAL EULERIAN PROGRAM IN TWO SPACE DIMENSIONS AND TIME,

SYSTEMS, SCIENCE AND SOFTWARE
P. O. BOX 1620
LA JOLLA, CALIFORNIA 92037

(10) Laura J. Hageman, Darin E. Wilkins, Robert T. Sedgwick
Jerry L. Waddell

(11) APR 76

(12) 170p.

(9) FINAL REPORT. JUL 74—JUN 75.

(15) F08635-75-C-0044

(16) 2560 (17) 02

AIR FORCE ARMAMENT LABORATORY
AIR FORCE SYSTEMS COMMAND • UNITED STATES AIR FORCE
EGLIN AIR FORCE BASE, FLORIDA

UNCLASSIFIED

SECURITY CLASSIFICATION OF THIS PAGE *(When Data Entered)*

REPORT DOCUMENTATION PAGE		READ INSTRUCTIONS BEFORE COMPLETING FORM
1. REPORT NUMBER AFATL-TR-76-45 ✓	2. GOVT ACCESSION NO.	3. RECIPIENT'S CATALOG NUMBER
4. TITLE *(and Subtitle)* HELP - A MULTIMATERIAL EULERIAN PROGRAM IN TWO SPACE DIMENSIONS AND TIME, BOOKS 1 AND 2		5. TYPE OF REPORT & PERIOD COVERED Final Report July 1974 to June 1975 6. PERFORMING ORG. REPORT NUMBER
7. AUTHOR(s) Laura J. Hageman Jerry L. Waddell Darin E. Wilkins Robert T. Sedgwick		8. CONTRACT OR GRANT NUMBER(s) F08635-75-C-0044 ✓
9. PERFORMING ORGANIZATION NAME AND ADDRESS Systems,,Science and Software P O Box 1620 La Jolla, California 92037		10. PROGRAM ELEMENT, PROJECT, TASK AREA & WORK UNIT NUMBERS 25600232
11. CONTROLLING OFFICE NAME AND ADDRESS Air Force Armament Laboratory (DLJW) Armament Development and Test Center Eglin Air Force Base, Florida 32542		12. REPORT DATE April 1976 13. NUMBER OF PAGES 437
14. MONITORING AGENCY NAME & ADDRESS*(if different from Controlling Office)*		15. SECURITY CLASS. *(of this report)* UNCLASSIFIED 15a. DECLASSIFICATION DOWNGRADING SCHEDULE

16. DISTRIBUTION STATEMENT *(of this Report)*

Distribution limited to U. S. Government agencies only; this report documents test and evaluation; distribution limitation applied April 1976. Other requests for this document must be referred to the Air Force Armament Laboratory (DLJW), Eglin Air Force Base, Florida 32542.

17. DISTRIBUTION STATEMENT *(of the abstract entered in Block 20, if different from Report)*

18. SUPPLEMENTARY NOTES

Available in DDC.

19. KEY WORDS *(Continue on reverse side if necessary and identify by block number)*

HELP
Multimaterial Interaction
Elastic-Plastic
Massless Tracer Particles
Continuum Mechanics
Hydrodynamic Codes RPM, OIL, and PIC

20. ABSTRACT *(Continue on reverse side if necessary and identify by block number)*

HELP is a two-dimensional, multimaterial, Eulerian Code for solving material flow problems in the hydrodynamic and elastic-plastic regimes. Although the code is basically Eulerian, material interfaces and free surfaces are propagated through the Calculation Mesh, in a Lagrangian manner, as discrete interfaces across which the materials are not allowed to interdiffuse. This interface treatment gives HELP a distinct advantage over other pure Eulerian Codes and allows it to be applied to the solution of a variety of complex, multimaterial problems. The basic HELP techniques have been exercised over a wide range of

DD FORM 1 JAN 73 1473 EDITION OF 1 NOV 65 IS OBSOLETE

UNCLASSIFIED

SECURITY CLASSIFICATION OF THIS PAGE *(When Data Entered)*

ITEM 20 (CONCLUDED):

solutions including hypervelocity and ballistic impact, fragmentation munitions, shaped charge jet formation, and shaped charge jet penetration. Considerable confidence has been gained in the basic numerical model.

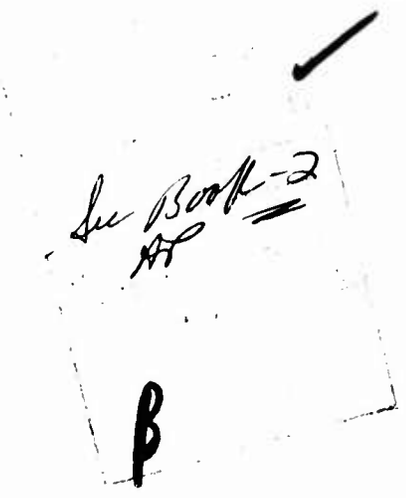
PREFACE

This report documents work performed during the period July 1974 through June 1975 by Systems, Science and Software, P. O. Box 1620, La Jolla, California 92036, under contract F08635-75-C-0044 with the Air Force Armament Laboratory, Armament Development and Test Center, Eglin Air Force Base, Florida 32542. Mr Leonard L. Wilson (DLDG) managed the program for the Armament Laboratory.

This technical report has been reviewed and is approved for publication.

FOR THE COMMANDER:

GERALD P. D'ARCY, Colonel, USAF
Chief, Guns, Rockets & Explosives Division



(The reverse of this page is blank)

CONTENTS

CONTENTS

CONTENTS

CONTENTS

CONTENTS

CONTENTS

CONTENTS

CONTENTS

CHAPTER I

INTRODUCTION

This is the second complete documentation of the HELP[1] code since it was developed five years ago. For those unfamiliar with HELP, the following remarks will give some historical perspective. The HELP code evolved from three major hydrodynamic codes developed over the past twenty years: RPM, OIL, and PIC. Starting from the one-material code, RPM[2], the HELP code added the capability of describing multimaterial interactions and of treating material strength as elastic-plastic rather than rigid-plastic. RPM, in turn, was an extension of OIL[3], a compressible fluid code, and included material strength in the rigid-perfectly plastic approximation. The OIL code was derived from the PIC[4] code and replaced the discrete PIC particles by a continuum. This latter transition to a continuum calculation was made in order to permit the treatment of very small compressions without having to use an extremely large number of particles for adequate mass resolution. The disadvantage of this transition, however, was the loss of certain Lagrangian-type features of PIC, and specifically the ability to treat flows containing more than one material. The major purpose of developing the HELP code was to restore the multimaterial capability of PIC while retaining the ease of treating small compressions and the computing economy of a continuous Eulerian model.

The accurate treatment of material interface or free surface motion, however, requires a detailed knowledge of surface location, which is not contained in a purely Eulerian model. The developers of the HELP code, therefore, deviated from the pure Eulerian model by employing massless tracer particles to track material interfaces and free surfaces.

These particles are initially placed along the surface boundaries and are moved thereafter with a local material velocity, thus creating a Lagrangian-type definition of moving surfaces without sacrificing the capability of easily treating extreme distortions. Furthermore, diffusion of materials across these surfaces is prevented in HELP by taking into account the surface positions when transporting material across the Eulerian cell boundaries.

In the years since HELP was first developed and documented it has been applied to a wide variety of multimaterial problems in continuum mechanics. In the process it has been significantly expanded to increase its generality and applicability. For example, sliplines were added in order to model more accurately the plugging failure in thin targets[5,6] and the high-explosive metal interactions in shaped charges[7] and fragmenting munitions.[8] Furthermore, a high explosive detonation model was added which would account for multiple explosives, multiple detonation points and wave shapers.[9]

As the applications of the HELP code increased the need for a more general problem generator grew. The original problem generator was written primarily for calculations involving spheres and cylinders impacting finite and semi-infinite plates. This original generator clearly was inadequate for generating shaped charge problems or even projectiles with various shaped noses. The new HELP generator can, without modification, generate any configuration where the package boundaries can be decomposed into straight line segments or arcs of circles and ellipses.

In addition to these improvements, some aspects of the code have been reformulated. For example, the hydrodynamic phase (HPHASE) has been simplified (the equations are solved in a single pass) in order to provide more accurate definitions of pressures and velocities near free surfaces. (See Section 2.2.2.) This has eliminated the need

to "glue" free surface cells, a method which proved unsatisfactory in certain situations. The single pass formulation also simplified the addition of an artificial viscosity term to cell boundary pressures. Another change involved the ordering of the three phases. It has been found that more realistic strain rates are computed using the post-transport cell velocities. Therefore the strength phase now precedes the hydrodynamic phase in the computational cycle. This reordering of the phases also led to their renaming: PH3 has become SPHASE; PH1 has become HPHASE, and PH2 has become TPHASE. And finally, the INFACE subroutine has been broken down into five subroutines, thereby reducing the core memory requirements of the code when it is segmented, and, perhaps more importantly, clarifying the use and treatment of the material interfaces.

A summary of the major chapters of the text follows. Chapter II gives a general description of the numerical method and the material models employed by HELP. The grid boundary conditions are discussed in detail in Chapter III as they apply to each phase of the calculation. Chapter IV offers a discussion of the material interfaces, as well as the determination of the pressure and shear yield strength for multimaterial cells. The equations which govern the slipline cells are given in Chapter V along with general guidelines on their use, and in Chapter VI the mechanisms added to the code for modelling plugging failure are described. Procedures for generating and restarting a HELP calculation are given in Chapter VII as well as some general statements about the code's capabilities and limitations. The interactive options available to the user are described in Chapter VIII, and a list of the error conditions are discussed in Chapter IX. In Chapter X a general flow diagram of the code, a description of all the subroutines, and a dictionary of variables is given. Several sample input decks for the

various types of problems to which HELP is commonly applied are given in Chapter XI. The HELP output is also described in this chapter. The last chapter, XII, describes several applications of the HELP code. The three appendices give instructions for dimensioning the arrays, an input form with abbreviated instructions for generating initial conditions, and a diagram indicating how the code might be segmented to save core storage.

This second documentation of the HELP code is considerably more detailed than the original and is more oriented toward the new user of the code. As with any large, complex computer code, HELP is most effectively used by those persons who have a good grasp of its numerical methods, material models, conventions, assumptions, capabilities and limitations. It is hoped by the present authors that this document will provide this basic information. Any suggestions for improvements in the code or its documentation will be appreciated by the authors.

CHAPTER II

GENERAL DESCRIPTION OF THE NUMERICAL METHOD

In the present section the conservation equations and the finite difference analog to treat these equations are given. The material model, which includes strength effects, is also presented.

2.1 CONSERVATION EQUATIONS

Space is divided into fixed Eulerian cells through which the material moves. To arrive at expressions for the rate of change of total mass, momentum and energy within such a cell, it is convenient to start with the conservation equations governing the motions and interactions of continuous media in the form:

$$\text{(Continuity Equation)} \qquad \frac{\partial \rho}{\partial t} = - \frac{\partial}{\partial x_i} (\rho u_i) \qquad (2.1)$$

$$\text{(Equation of Motion)} \qquad \rho \frac{Du_j}{Dt} = \frac{\partial}{\partial x_i} (\sigma_{ij}) \qquad (2.2)$$

$$\text{(Energy Equation)} \qquad \rho \frac{DE_T}{Dt} = \frac{\partial}{\partial x_i} (\sigma_{ij} u_j) \quad . \qquad (2.3)$$

Here σ_{ij} is the stress tensor, which can be regarded as the sum of the hydrostatic stress, $-\delta_{ij}P$, and a stress deviator tensor, s_{ij}, i.e.,

$$\sigma_{ij} = s_{ij} - \delta_{ij}P \qquad (2.4)$$

and $E_T = \frac{1}{2} u_i u_i + E_I$ is the total energy (kinetic plus internal) per unit mass. Tensor notation is implied, so that repeated indices denote summations.

Expanding the convective derivatives in Eqs. (2.2) and (2.3), $Df/Dt = \partial f/\partial t + u_i \, \partial f/\partial x_i$, then adding Eq. (2.1) times u_j to Eq. (2.2), and Eq. (2.1) times E_T to Eq. (2.3), and collecting terms, gives

$$\frac{\partial}{\partial t} (\rho u_j) = \frac{\partial}{\partial x_i} \sigma_{ij} - \frac{\partial}{\partial x_i} (\rho u_i u_j) \tag{2.5}$$

$$\frac{\partial}{\partial t} (\rho E_T) = \frac{\partial}{\partial x_i} (\sigma_{ij} u_j) - \frac{\partial}{\partial x_i} (\rho u_i E_T) \quad . \tag{2.6}$$

For the developments to follow it is desirable to replace these differential equations by the analogous integral equations, obtained by integrating over the cell volume, V, and then converting the volume integral of divergences to surface integrals over the cell surfaces. Equations (2.1), (2.5), and (2.6) then become

$$\frac{\partial}{\partial t} \int_V \rho dV = - \int_S \rho u_i n_i dS \tag{2.7}$$

$$\frac{\partial}{\partial t} \int_V \rho u_j dV = \int_S \sigma_{ij} n_i dS - \int_S \rho u_i u_j n_i dS \tag{2.8}$$

$$\frac{\partial}{\partial t} \int_V \rho E_T dV = \int_S \sigma_{ij} u_j n_i dS - \int_S \rho u_i E_T n_i dS \quad . \tag{2.9}$$

2.2 DIVISION INTO PHASES

It is convenient to express the integral conservation relations, Eqs. (2.7) through (2.9), as finite difference equations over the time step Δt^* and also to decompose the total stress, σ_{ij}, into its deviator and hydrostatic components, according to Eq. (2.4). This gives, for the increments of total mass (m), momenta (mu_j) and energy (mE_T) within the cell,

$$\Delta m = - \Delta t \int_S \rho u_i n_i dS \tag{2.10}$$

$$\Delta(mu_j) = \Delta t \int_S s_{ij} n_i dS - \Delta t \int_S P n_j dS - \Delta t \int_S (\rho u_i u_j) n_i dS \tag{2.11}$$

$$\Delta(mE_T) = \Delta t \int_S s_{ij} u_j n_i dS - \Delta t \int_S P u_i n_i dS - \Delta t \int_S (\rho u_i E_T) n_i dS . \tag{2.12}$$

Here, the terms on the right are divided into increments due to stress deviator forces on the cell surface (first column), those due to the pressure forces on the cell surface (second column), and those due to the transport of mass, momentum and energy through the surface of the cell (third column). These three types of increments are accounted for in distinct phases of the computation. Specifically, during each time step all cells are updated three times to account for the following:

- Effects of the stress deviators (SPHASE, Strength Phase),
- Effects of pressure (HPHASE, Hydrodynamic Phase),
- Effects of transport (TPHASE, Transport Phase).

The phases are calculated in the order listed. In the discussion that follows, the calculation of the terms on the right of Eqs. (2.10) through (2.12) are described sequentially starting with SPHASE. The equations are written in cylindrical

---

* See DT in Section 10.3.

coordinates although the code models both the axisymmetric and plane strain cases.

Some preliminary definitions will be useful. Superscript n on a variable refers to the value of the variable at the beginning of the time step and superscript (n+1) denotes the value at the end. In this discussion a typical cell in the interior of the grid is considered; Chapter III has a discussion of the special conditions which describe the calculation at the grid boundaries or at the axis of symmetry. For a typical cell, denoted by a value of the index k, the dependent variables for that cell are written P(k), u(k), v(k), E_I(k), m(k), representing respectively the pressure, radial and axial components of velocity, the specific internal energy and the mass for cell k. The adjacent cells above, below, to the right and left of k will be designated respectively as ka, kb, kr and kℓ. Here the terms above, below, right and left refer to a cross-section view of the cells, in which the left border is parallel to the axis of symmetry with z increasing upward (see Figure 2.1). Each cell is, then, the torus obtained by rotating the rectangle (since $\Delta r \neq \Delta z$ in general) about the axis of symmetry.

2.2.1 SPHASE – The Effects of Deviatoric Stresses

This section discusses the modification of the cell velocity and internal energy due to the stress deviators. The derivation of the stress deviators is given in Section 2.3.2. The SPHASE subroutine should be executed only if at least one material in the problem has shear yield strength.

2.2.1.1 Continuity Equation, (2.10)

No contribution in SPHASE.

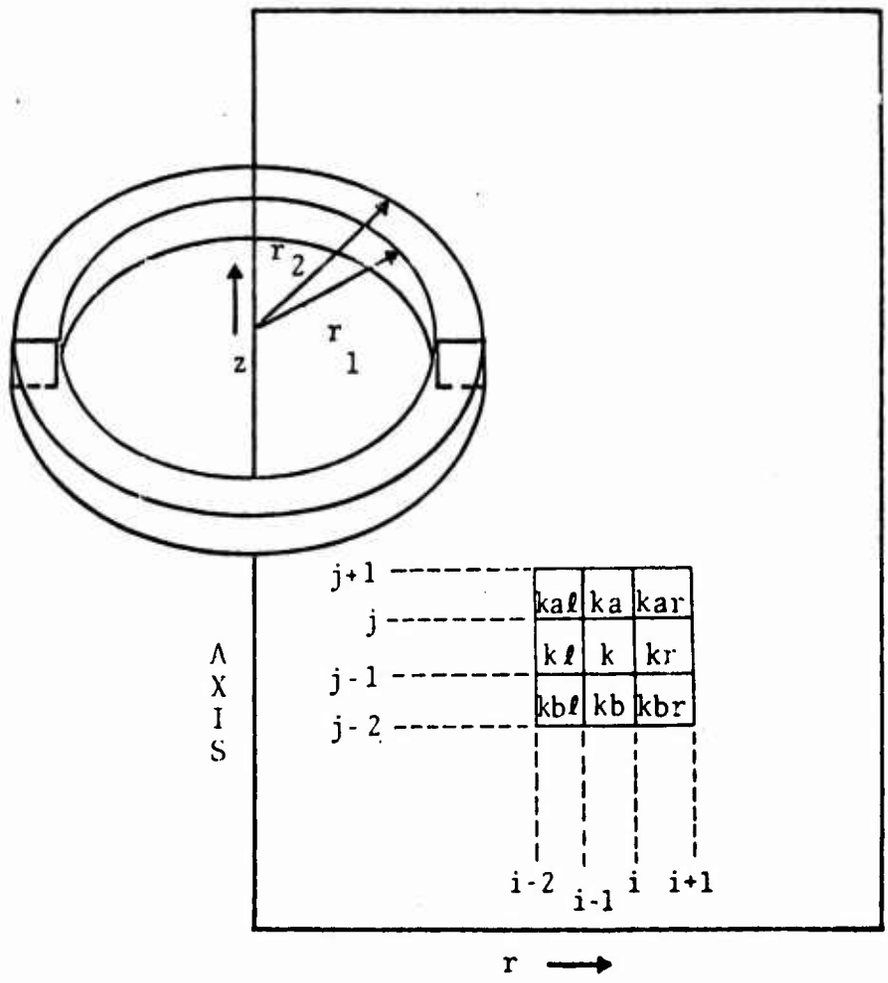


Figure 2.1--Grid layout and typical cell in cylindrical coordinates.

2.2.1.2 Equation of Motion, (2.11)

The SPHASE contribution to the equation of motion (2.11) is

$$\Delta_s(mu_j) = \Delta t \int_S s_{ij} n_i dS \quad .$$

The axial and radial momentum increments will be discussed separately.

Effect on Axial Motion - In this case the velocity u_j is the axial velocity, and the term $s_{ij}n_i$ is the axial component of the deviatoric stress on a surface. For a torus with rectangular section, $s_{ij}n_i$ on the various faces is

$$s_{zz}^{(t)} \quad \text{cell top}$$

$$- s_{zz}^{(b)} \quad \text{cell bottom}$$

$$s_{rz}^{(r)} \quad \text{outer cell surface}$$

$$- s_{rz}^{(\ell)} \quad \text{inner cell surface}$$

where zz and rz subscripts denote normal and shear stresses. The top, bottom, right and left surfaces are indicated by (t), (b), (r), and (ℓ), respectively. The equation for axial motion is therefore

$$\Delta_s(mv) = \left(s_{zz}^{(t)} - s_{zz}^{(b)}\right) \pi \left(r_2^2 - r_1^2\right) \Delta t + s_{rz}^{(r)} \, 2\pi r_2 \, \Delta z \, \Delta t$$

$$- s_{rz}^{(\ell)} \, 2\pi r_1 \, \Delta z \, \Delta t \quad ,$$

where r_1 and r_2 are the inner and outer radii, respectively, of the cell and Δz is its axial dimension.

In this equation, and in the remainder of this section, cell boundary stresses are determined from a simple average of the time n cell centered stresses, e.g.,

$$s_{zz}^{(t)} = \frac{s_{zz}^{n}(k) + s_{zz}^{n}(ka)}{2}$$

$$s_{rz}^{(r)} = \frac{s_{rz}^{n}(k) + s_{rz}^{n}(kr)}{2} \quad .$$

Effect on Radial Motion

In this case the velocity u_j is the radial velocity, and the radial components $s_{ij}n_i$ of the deviatoric stresses on the various faces of the mass element as shown in Figure 2.2 are

$s_{zr}^{(t)}$ top of mass element

$-s_{zr}^{(b)}$ bottom

$s_{rr}^{(r)}$ right

$-s_{rr}^{(\ell)}$ left

$-s_{\theta\theta}(\Delta\theta/2)$ front

$-s_{\theta\theta}(\Delta\theta/2)$ back .

The mass of the element is $m\Delta\theta/2\pi$, where m is the cell mass,

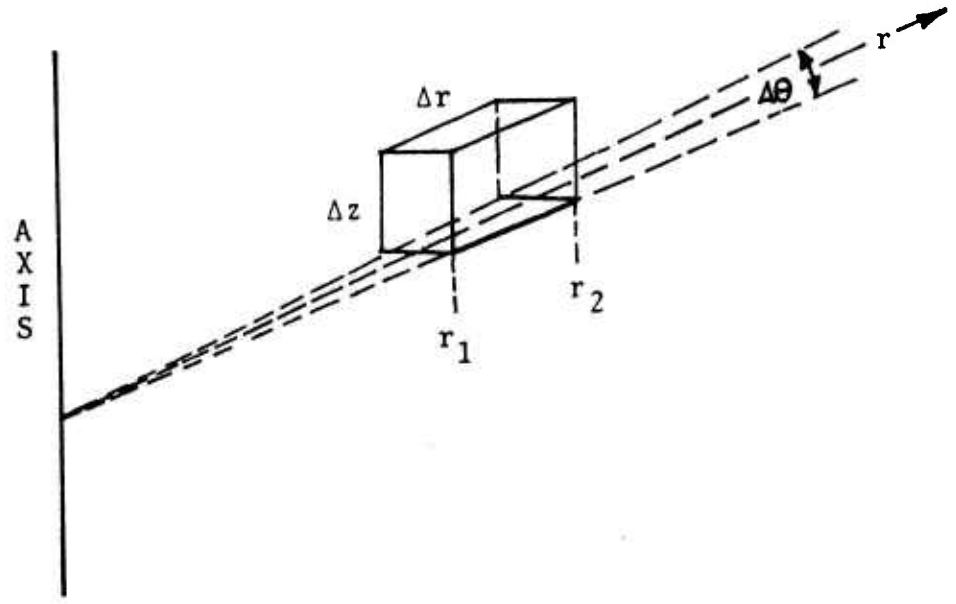


Figure 2.2--Element of volume for discussion of radial motion in cylindrical coordinates.

and

$$\frac{\Delta\theta}{2\pi}\,\Delta_s(mu) = \left(s_{zr}^{(t)} - s_{zr}^{(b)}\right)\left(r_2^2 - r_1^2\right)\frac{\Delta\theta}{2}\,\Delta t + s_{rr}^{(r)}\, r_2\,\Delta\theta\,\Delta z\,\Delta t$$

$$- s_{rr}^{(\ell)}\, r_1\Delta\theta\,\Delta z\,\Delta t - s_{\theta\theta}\Delta z\Delta r\,\Delta\theta\,\Delta t$$

or (multiplying by $2\pi/\Delta\theta$) the equation for radial motion is

$$\Delta_s(mu) = 2\pi\,\Delta t\left[\frac{1}{2}\left(s_{zr}^{(t)} - s_{zr}^{(b)}\right)\left(r_2^2 - r_1^2\right)\right.$$

$$\left. + s_{rr}^{(r)}\, r_2\,\Delta z - s_{rr}^{(\ell)}\, r_1\,\Delta z - s_{\theta\theta}\Delta z\,\Delta r\right] .$$

2.2.1.3 Energy Equation, (2.12)

In the SPHASE term from Eq. (2.12),

$$\Delta_s(mE_T) = \Delta t\int_S s_{ij}u_j n_i\,dS \quad ,$$

$s_{ij}u_j n_i$ is the work rate per unit surface area which, for the various faces of the torus, is

$$\left(s_{zz}^{(t)}v^{(t)} + s_{zr}^{(t)}u^{(t)}\right) \quad \text{cell top}$$

$$-\left(s_{zz}^{(b)}v^{(b)} + s_{zr}^{(b)}u^{(b)}\right) \quad \text{cell bottom}$$

$$\left(s_{rz}^{(r)}v^{(r)} + s_{rr}^{(r)}u^{(r)}\right) \quad \text{outer cell surface}$$

$$-\left(s_{rz}^{(\ell)}v^{(\ell)} + s_{rr}^{(\ell)}u^{(\ell)}\right) \quad \text{inner cell surface}$$

where (t), (b), (r), (ℓ) denotes stresses and velocities at the top, bottom, right and left cell faces, respectively. The

cell face velocities are determined from a simple average of the time n velocities in the two cells, e.g.,

$$v^{(t)} = \frac{v^n(k) + v^n(ka)}{2}$$

$$u^{(\ell)} = \frac{u^n(k) + u^n(k\ell)}{2} \quad .$$

The equation for the SPHASE change in total cell energy is

$$\Delta_s(mE_T) = \left(s_{zz}^{(t)}v^{(t)} + s_{zr}^{(t)}u^{(t)} - s_{zz}^{(b)}v^{(b)} - s_{zr}^{(b)}u^{(b)}\right)\pi\left(r_2^2 - r_1^2\right)\Delta t$$

$$+ \left(s_{rz}^{(r)}v^{(r)} + s_{rr}^{(r)}u^{(r)}\right) 2\pi\, r_2\, \Delta z\, \Delta t$$

$$- \left(s_{rz}^{(\ell)}v^{(\ell)} + s_{rr}^{(\ell)}u^{(\ell)}\right) 2\pi\, r_1\, \Delta z\, \Delta t \quad .$$

The post-SPHASE cell specific kinetic energy, $1/2\ u_i u_i$, is determined by the post-SPHASE velocities obtained from the equation of motion. The post-SPHASE specific internal energy for the cell, E_I, is then calculated from the updated value of specific total energy and specific kinetic energy,

$$E_I = E_T - 1/2\ u_i u_i \quad .$$

The partitioning of the internal energy increment among the various materials in a multimaterial cell is discussed in Section 4.6.3.

This completes the discussion of the momentum and energy increments due to the deviator stresses as calculated in SPHASE.

2.2.2 HPHASE – The Effects of Pressure

This section discusses the modification of the cell velocity and internal energy due to hydrostatic pressures. The discussion will be in terms of pressures acting on cell boundaries. Sections 2.2.2.4 and 2.2.2.5 will discuss the definitions of these cell boundary pressures.

2.2.2.1 Continuity Equation, (2.10)

No contribution in HPHASE.

2.2.2.2 Equation of Motion, (2.11)

The HPHASE contribution to the equation of motion (2.11) is

$$\Delta_H(mu_j) = -\Delta t \int_S P\, n_j dS \quad .$$

The axial and radial momentum increments will be discussed separately.

Effect on Axial Motion

In this case u_j is the axial velocity and $n_j = -1, 0, +1$ are the axial components of the unit normal to the bottom, sides and top of the cell, respectively. The equation of axial motion is therefore

$$\Delta_H(mv) = P^{(b)}\pi\left(r_2^2 - r_1^2\right)\Delta t - P^{(a)}\pi\left(r_2^2 - r_1^2\right)\Delta t$$

where r_2 and r_1 are the radii of the outer and inner cell surfaces, respectively, and $P^{(b)}$, $P^{(a)}$ are the pressures on the bottom and top cell faces.

Effect on Radial Motion

In this case the velocity u_j is the radial velocity. Referring to Figure 2.2, the radial components of the unit normal take the values $n_j = -1, 1, -\Delta\theta/2, -\Delta\theta/2, 0, 0$, corresponding to the inner and outer surfaces, the two side surfaces and the top and bottom, respectively. The pressure-integral contribution in Equation (2.11) becomes

$$\frac{\Delta\theta}{2\pi}\,\Delta_H(mu) = P^{(\ell)} r_1 \Delta\theta\ \Delta z\ \Delta t - P^{(r)} r_2 \Delta\theta\ \Delta z\ \Delta t + P^{(s)} \Delta r\ \Delta z\ \Delta\theta\ \Delta t$$

where $P^{(\ell)}$, $P^{(r)}$ and $P^{(s)}$ are the left, right and side face pressures, respectively. The side pressure $P^{(s)}$ is taken to be the average of the left and right boundary pressures

$$P^{(s)} = \frac{P^{(\ell)} + P^{(r)}}{2} .$$

Since $\Delta r = r_2 - r_1$ the equation for radial motion becomes

$$\Delta_H(mu) = \left(P^{(\ell)} - P^{(r)}\right)\left(\frac{r_1 + r_2}{2}\right) 2\pi\Delta z\Delta t \quad .$$

2.2.2.3 Energy Equation, (2.12)

In the HPHASE term from Equation (2.12),

$$\Delta_H(mE_T) = -\Delta t \int_S Pu_i n_i dS \quad ,$$

$Pu_i n_i$ is the work rate per unit surface area which, for the various faces of the torus, is

$p^{(a)}{}_{v}{}^{(a)}$	cell top
$-p^{(b)}{}_{v}{}^{(b)}$	cell bottom
$p^{(r)}{}_{u}{}^{(r)}$	outer cell surface
$-p^{(\ell)}{}_{u}{}^{(\ell)}$	inner cell surface

where the superscripts a, b, r, ℓ denote pressures and velocities at the top, bottom, right and left cell faces, respectively. Multiplying these terms by the time step and by the surface area they are acting upon gives the HPHASE equation for the change in cell total energy as

$$\Delta_H(mE_T) = -\Delta t\,[(p^{(a)}{}_{v}{}^{(a)} - p^{(b)}{}_{v}{}^{(b)})\,\pi\,(r_2^2 - r_1^2) + (p^{(r)}{}_{u}{}^{(r)}{}_{r_2} - p^{(\ell)}{}_{u}{}^{(\ell)}{}_{r_1})\,2\pi\Delta z]\ .$$

The post-HPHASE cell specific kinetic energy, $1/2\ u_i u_i$, is determined by the post-HPHASE velocities obtained from the equation of motion. The post-HPHASE specific internal energy, E_I, is then calculated from the updated value of specific total energy and specific kinetic energy:

$$E_I = E_T - \frac{1}{2}\,u_i u_i\ .$$

The partitioning of the updated internal energy among materials in multimaterial cells is discussed in Section 4.6.3.

2.2.2.4 Treatment of Free Surfaces and Large Density Discontinuities

Thus far, the HPHASE equations for momentum and energy increments have been derived in terms of cell boundary pressures and velocities. Normally, the cell boundary values are defined to be the simple average of the post-SPHASE cell centered quantities on either side of the boundary, e.g.,

$$P^{(b)} = \frac{P(k) + P(kb)}{2}$$

$$P^{(a)} = \frac{P(k) + P(ka)}{2}$$

$$u^{(r)} = \frac{u(k) + u(kr)}{2}$$

$$u^{(\ell)} = \frac{u(k) + u(k\ell)}{2} \quad .$$

However, these definitions can lead to excessive acceleration of partially-filled, free-surface cells or of cells containing relatively low density material. In the case of a full cell adjacent to a partially-filled, free-surface cell, it would seem that the condition at the boundary between them should relate more closely to the void than the normal averaging procedure would allow. The over-acceleration of the partially-filled cells would be prevented by reducing the pressure gradient acting on them.

One method of requiring a boundary pressure to reflect the presence of voids and low-density materials more closely is to define the cell boundary pressure to be an inverse compression weighted average of the adjacent cells, i.e., for cells k and kb the boundary pressure $P^{(b)}$ between them is defined to be

$$P^{(b)} = \frac{C(kb)P(k) + C(k)P(kb)}{C(kb) + C(k)}$$

where C(k) and C(kb) are defined (for pure cells and free-surface cells having only one material) as follows:

$$C(k) = [m(k)/VOL(k)]/\rho_o$$

$$C(kb) = [m(kb)/VOL(kb)]/\rho_o$$

where m(k), m(kb), VOL(k), VOL(kb), are the total masses and total volumes of their respective cells, and ρ_0 is the reference density. The above compression definitions are meant to be used as weighting factors, and, in the case of one material free-surface cells, assume the material is smeared over the entire volume of the cell. For multimaterial cells the compression term is a simple average of the compressions of the materials in the cell.

The effect of inverse compression weighting is to weigh more heavily the pressure associated with the underdense cell when defining the cell boundary pressure. This lowers the pressure gradient acting on the underdense cell. It should be noted that if the neighboring cells have the same compression, inverse compression weighting reduces to the simple averaging procedure used before.

For boundaries where inverse compression weighting is used to determine the cell boundary pressures, the cell boundary velocities $u^{(b)}$ and $v^{(b)}$ used in the energy equation are defined to be a compression weighted average, e.g.,

$$u^{(b)} = \frac{C(k)u(k) + C(kb)u(kb)}{C(k) + C(kb)}$$

$$v^{(b)} = \frac{C(k)v(k) + C(kb)v(kb)}{C(k) + C(kb)} \quad .$$

This definition weighs more heavily the velocity of the denser cell and assumes its velocity is probably more realistic than the velocity of the less dense cell.

The process of inverse compression weighting for boundary pressures and compression weighting for boundary velocities is used at all boundaries of a free-surface cell. However, since the over-acceleration problem can occur whenever a dense, high-pressure cell is adjacent to an underdense,

low-pressure cell, an input variable, CRATIO, has been added to the code in order to allow the user to control the use of the compression weighting. Whenever the compression ratio between two cells is greater than CRATIO, the pressure at the boundary between them is taken to be the inverse compression weighted average of the cell-centered values, and the boundary velocity is taken to be the compression weighted average. CRATIO has a default value of 10^4 which essentially restricts the use of the weighted averages to the free surface.

2.2.2.5 Artificial Viscosity

The stability of a HELP calculation depends, in part, on the effective viscosity which results from converting kinetic energy into internal energy during mass transport. If velocities are small, there is little mass flux between cells, and as a result, little effective viscosity. Since these conditions have arisen in a variety of calculations, an option, LVISC, has been added which, when set equal to one, will signal the code to add an artificial viscosity term to the cell boundary pressures in subroutine HPHASE.

The discussion which follows presents a form of artificial viscosity which has been useful in several calculations. However, when the code is applied to a new class of problems, it may be that other artificial viscosity formulations will give improved results. The user can make such changes if the need arises.

The definition of the artificial viscosity, $Q^{(a)}$, which can be added to $P^{(a)}$, the pressure at the top cell boundary is

$$Q^{(a)} = \left(v(k) - v(ka) \right) \cdot \sqrt{\left| \frac{P(k) + P(ka)}{2} \right| \cdot \frac{\rho(k) + \rho(ka)}{2}}$$

where ρ is material density.

An analogous term, $Q^{(b)}$, added to $P^{(b)}$, the pressure at the bottom cell boundary, can be defined by substituting kb for ka in the above equation.

For the right cell boundary, the artificial viscosity is

$$Q^{(r)} = \left(u(k) - u(kr) \right) \cdot \sqrt{\left| \frac{P(k) + P(kr)}{2} \right| \cdot \frac{\rho(k) + \rho(kr)}{2}}$$

and an analogous term, $Q^{(\ell)}$, is defined for the left cell boundary by substituting kℓ for kr.

If one of the cell boundaries is also a transmittive grid boundary, the artificial viscosity term for that cell boundary is zero. If, instead, one of the cell boundaries is a reflective grid boundary, the artificial viscosity for that cell boundary is

$$-2 \cdot \omega(k) \sqrt{|P(k)| \cdot \rho(k)} \quad ,$$

where ω is the velocity component normal to that boundary.

2.2.3 TPHASE - The Effects of Transport

The purpose of this section is to describe how the transport of mass, momentum and energy from cell to cell is accounted for in the code. This is done by calculating the integrals in the last terms of Eqs. (2.10) through (2.12). In the discussion below, attention is given to the case of transport between pure cells. The special provisions required to treat interface cells are described in Chapter IV.

2.2.3.1 Continuity Equation, (2.10)

From Eq. (2.10) the mass being transported is

$$\Delta_T(m) = - \Delta t \int_S \rho u_i n_i dS$$

and is determined for each cell face from

$$\delta m = - \rho^d \bar{u}_i A_i \Delta t \quad ,$$

where ρ^d is the density of the cell from which the mass moves (donor cell), A_i is the area of the cell face and $\bar{u}_i$ is an interpolated value of the velocity component normal to the cell face and represents the velocity at the interface at the end of the time step. For example, considering transport from cell k into the cell above it, ka, the cell boundary velocity, $v^{(a)}$, is defined to be a simple average of the velocities of the two cells. However, we want the mass transport velocity, $\bar{v}$, to be the cell boundary velocity at the end of the time step, Δt. This transport velocity therefore is the velocity of the material located a depth $\Delta z = \bar{v}\Delta t$ into the donor cell k (see Figure 2.3). However, since the transport velocity $\bar{v}$, when adjusted by the axial velocity gradient, must equal the cell boundary velocity, we have

$$\bar{v} + \frac{\partial v}{\partial z} \Delta z = v^{(a)} \quad ,$$

or

$$\bar{v} \left(1 + \frac{\Delta v}{\Delta z} \Delta t\right) = v^{(a)} \quad .$$

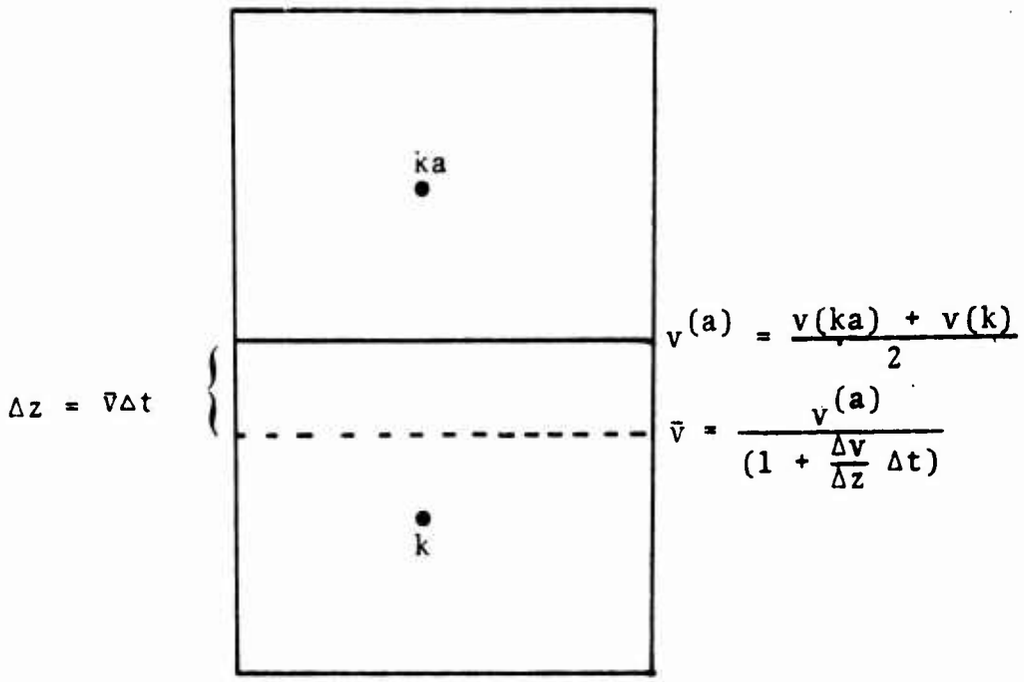


Figure 2.3 --The relationship between $v^{(a)}$, the cell boundary velocity at the beginning of TPHASE, and $\bar{v}$, the transport velocity.

Thus the axial mass transport velocity is defined to be

$$\bar{v} = \frac{\frac{1}{2}\left[v(k) + v(ka)\right]}{\left[1 + \frac{v(ka) - v(k)}{\Delta z}\,\Delta t\right]} \quad .$$

Similarly, the radial mass transport velocity between the donor cell k and its neighbor on the right, kr, is defined to be

$$\bar{u} = \frac{\frac{1}{2}\left[u(k) + u(kr)\right]}{\left[1 + \frac{u(kr) - u(k)}{\Delta r}\,\Delta t\right]} \quad .$$

Calculated transport masses are subtracted from the donor cell mass and added to the acceptor cell mass, and the cell momenta and internal energy are updated accordingly. Since all transport terms are computed using the post-HPHASE quantities, a given cell is updated only after the transport terms have been calculated for all four of its boundaries.

2.2.3.2 Equation of Motion, (2.11)

The TPHASE contribution from the equation of motion (2.11) is

$$\Delta_T(mu_j) = -\,\Delta t \int_S (\rho u_i u_j) n_i dS \quad .$$

The axial and radial momentum increments will be discussed separately.

Effect on Axial Motion

At each of the four cell faces comprising the surface of the cell, the specific axial momentun, u_j, being transported is the axial velocity of the cell from which the mass moves, i.e.,

$$u_j = v(kd)$$

where kd is the index of the donor cell. Since this velocity is constant over the cell face it can be removed from the integral sign and we obtain

$$\Delta_T(mv) = v(kd) \left[-\Delta t \int_S \rho u_i n_i dS\right] \quad .$$

The quantity in brackets is seen to be the mass transported through the cell face as calculated in Section 2.2.3.1. Thus for each cell face,

$$\Delta_T(mv) = v(kd)\ \delta m \quad .$$

The total axial momentum increment for the cell is the sum of the increments obtained for each of the four cell faces.

Effect on Radial Motion

The specific radial momentum, u_j, being transported is the radial velocity of the cell from which the mass moves, i.e.,

$$u_j = u(kd)$$

where kd is the index of the donor cell. By an analysis similar to that for the axial momentum increment, the radial momentum increment is

$$\Delta_T(mu) = u(kd)\ \delta m$$

for each of the cell faces. The total radial momentum increment for the cell is the sum of the increments obtained for each of the four cell faces.

2.2.3.3 Energy Equation, (2.12)

The expression for the transport of energy from Eq. (2.12) is

$$\Delta_T(mE_T) = -\Delta t \int_S (\rho u_i E_T)\, n_i dS \quad .$$

To evaluate this integral, the transported specific energy, E_T^d, across each cell face is taken to be that of the donor cell, kd, i.e.,

$$E_T^d = E_I(kd) + \frac{1}{2}\left[\left(u(kd)\right)^2 + \left(v(kd)\right)^2\right]$$

and the total energy which is transported across a given interface is therefore the product of this specific energy and the associated transport mass, which was computed above in the continuity equation, i.e.,

$$\Delta_T(mE_T) = E_T^d \delta m \quad .$$

The total energy increment for the cell is the sum of the increments obtained for each of the four cell faces. The internal energy is obtained by subtracting the new kinetic energy (obtained from the updated mass and momenta) from the updated total energy. The partitioning of the updated internal energy among materials in multimaterial cells is discussed in Section 4.6.3.

2.3 MATERIAL MODEL

As is well known, the conservation equations (Eqs. 2.1 through 2.3), which consist (in two dimensions) of four equations, are insufficient to allow determination of the eight unknowns (ρ, u, v, P, E_T, s_{rr}, s_{rz}, s_{zz}) contained in them and must be supplemented by equations defining the

material model to be used. The material model employed in the HELP code can represent inert materials or high explosives (HE) and provides a fairly sophisticated HE detonation routine allowing for the detonation of multiple HE packages and for detonation around inert obstacles. The code also includes strength effects, modeling them in the elastic-plastic regime, and a simple tensile failure model. By bypassing the strength effects subroutine, SPHASE, the code solves problems in compressible fluid mechanics. A more complete discussion of these points follows.

2.3.1 Equation of State

The HELP code employs three different equations of state: the Jones-Wilkins-Lee (JWL) equation of state[10] for high explosives, the Tillotson equation of state[11] for inert materials, and the ideal gas equation of state. The array dimensions of the HELP code allow up to 30 materials in any problem. The following convention is used to relate the 30 material numbers to one of the three equations of state: materials 1-19 are assumed to use the Tillotson equation of state; material 20 is assumed to be an ideal gas; and materials 21-30 are assumed to be high explosives and use the JWL equation of state.

2.3.1.1 High Explosives

Once an explosive cell is detonated, its pressure is computed using the Jones-Wilkins-Lee (JWL) equation of state.[10] The equation is

$$P(E_I,\rho) = A\left(1 - \frac{a\rho}{\alpha\rho_o}\right)e^{-\frac{\alpha\rho_o}{\rho}} + B\left(1 - \frac{a\rho}{\beta\rho_o}\right)e^{-\frac{\beta\rho_o}{\rho}} + a\rho E_I$$

where

E_I = specific internal energy in ergs/g
ρ = density of detonation products in g/cm^3
ρ_o = density of undetonated explosive in g/cm^3.

The constants ρ_o, A, B, a, α and β for four explosives are listed in Table 2.1. The constants for these four explosives are defined via DATA statements in EQST, and the material numbers of these explosives are: 21-Comp B; 22-TNT; 23-OCTOL; 24-PBX 9404.

The JWL equation of state has a pressure maximum which is potentially hazardous to the pressure iteration (see Section 4.6.1). The pressure iteration adjusts the densities of the materials in a multimaterial cell to achieve pressure equilibrium. During this iteration process, it is possible for the code to reference the JWL equation of state with a density corresponding to a pressure beyond this maximum, thereby preventing convergence of the iteration. In order to preclude this possibility, a variable, RHOMAX, is defined via DATA statements in subroutine CDT. This is the maximum value of density for each HE to be used in the pressure iteration. The values of RHOMAX for the four specified explosives are listed in Table 2.1.

The sound speed, C, in the detonated explosive is defined by the equation

$$C = \sqrt{\Gamma\ P/\rho} \quad .$$

The value of Γ (defined by DATA statements in subroutine CDT) as well as the detonation velocity, D, (defined by DATA statements in subroutine DETIME) and detonation energy, E_o, (defined by DATA statements in ADDENG) for the four named explosives are listed in Table 2.1.

TABLE 2.1

JWL EQUATION-OF-STATE PARAMETERS FOR FOUR EXPLOSIVES[1]

HELP Material Code Number	High Explosive	A (dynes/cm^2)	B (dynes/cm^2)	a	α	β	ρ_0 (g/cm^3)	Γ	E_o (ergs/g)	D (cm/sec)	RHOMAX (g/cm^3)
21	COMP B	5.2423×10^{12}	7.6780×10^{10}	0.34	4.2	1.10	1.717	2.706	4.95×10^{10}	7.98×10^{5}	9.0
22	TNT	3.7377×10^{12}	3.7471×10^{10}	0.35	4.15	0.90	1.630	2.727	3.681×10^{10}	6.93×10^{5}	9.5
23	OCTOL	7.4860×10^{12}	1.3380×10^{11}	0.375	4.5	1.20	1.821	2.830	5.271×10^{10}	8.48×10^{5}	9.5
24	PBX-9404	6.7202×10^{12}	2.0675×10^{11}	0.25	4.25	1.45	1.84	2.850	5.543×10^{10}	8.80×10^{5}	11.0

[1] Lee, E., M. Finger, and W. Collins, "JWL Equation of State Coefficients for High Explosives," Lawrence Livermore Laboratory, UCID-16189, January, 1973.

2.3.1.2 Inert Materials

The equation of state used in HELP, for initially condensed materials, is that due to J. H. Tillotson,[11] modified to give a smooth transition between condensed and expanded states. For the condensed states, i.e., when $\rho/\rho_o > 1$, or for any cold states, $E_I < E_s$, the equation has the form

$$P = P_C = \left[a + \frac{b}{\frac{E_I}{E_o \eta^2} + 1} \right] E_I \rho + A\mu + B\mu^2$$

where

E_I = specific internal energy in ergs/g

ρ = material density in g/cm^3

$\eta = \rho/\rho_o$

$\mu = \rho/\rho_o - 1$.

For expanded hot states, i.e., when $\rho/\rho_o < 1$ and $E_I > E_s'$ the equation of state has the form

$$P = P_E = aE_I\rho + \left[\frac{bE_I\rho}{\frac{E_I}{E_o \eta^2} + 1} + A\mu e^{-\beta(\rho_o/\rho - 1)} \right] e^{-\alpha(\rho_o/\rho - 1)^2} .$$

A smooth transition between the condensed and expanded states is insured by a transition equation for the intermediate region defined by $E_s < E_I < E_s'$ and $\rho/\rho_o < 1$. This blended portion of the equation of state has the form

$$P = \frac{(E_I - E_s)\, P_E + (E_s' - E_I)\, P_C}{E_s' - E_s} .$$

In these equations, a, b, α, β, E_o, E_s, E_s', A, B, and ρ_o are constants for the particular material. The values of these constants for nineteen materials are given in Table 2.2.

For multimaterial cells, an iteration is used to determine the cell pressure and the densities of the various materials within the cell. This iteration is discussed in Section 4.6.1.

2.3.1.3 Ideal Gas

As noted earlier, material number 20 is assumed to be an ideal gas and the pressure is calculated from the ideal gas equation of state

$$P = (\gamma-1)\ \rho\ E_I$$

and the sound speed given by

$$C = \sqrt{\gamma\ P/\rho}$$

with ρ being the density, E_I the specific internal energy, and γ the ideal gas constant. The normal density, ρ_o, of the ideal gas is defined in a DATA statement to be that of air at standard temperature and pressure (0.00129 g/cm^3) and is used only for editing purposes. Since γ is stored in a non-dimensioned variable, GAMMA, the code has the capability of handling only one ideal gas. Note, however, that the JWL equation of state reduces to the ideal gas equation of state if A = B = 0 and a = γ-1. Also, in order for a material using the JWL equation of state to have the ideal gas sound speed, it is necessary to have $\Gamma = \gamma$.

TABLE 2.2 HELP EQUATION OF STATE CONSTANTS

	HELP Material Code Number	Material	a	b	A dynes/cm^2	B dynes/cm^2	E_0 ergs/g	α	β	E_S ergs/g	ρ_o g/cm^3	E_S ergs/g
1	1	W	0.5	1.04	3.08×10^{12}	2.5×10^{12}	0.225×10^{12}	10	10	1.11×10^{10}	19.17	5.6×10^{10}
	2	Cu	0.5	1.5	1.39	1.1	0.325	5	5	1.38	8.9	6.9
	3	Fe	0.5	1.5	1.28	1.05	0.095	5	5	2.44	7.8	10.2
	4	Al	0.5	1.63	0.75	0.65	0.05	5	5	3.0	2.79	15.0
	5	Be	0.55	0.62	1.17	0.55	0.175	5	5	10.0	1.8	46.0
	6	Ti	0.5	0.60	1.03	0.5	0.07	5	5	3.5	4.5	12.5
	7	Ni	0.5	1.33	1.91	1.5	0.09	5	5	2.85	8.9	9.4
2	8	Mo	0.5	1.02	2.71	1.65	0.045	5	5	2.8	10.2	9.0
	9	Th	0.4	0.86	0.53	0.5	0.025	9	0.88	2.0	11.7	8.0
3	10	Pb	0.4	2.4	0.466	0.15	0.02	13	15	0.26	11.3	9.7
	11	CH_2	0.6	2.0	0.075	0.02	0.07	10	5	2.4	0.9	18.0
4	12	Granite[5]	0.5	1.3	0.60	0.00	0.16	5	5	3.5	2.7	18.0
	13	Andesite[5]	0.5	1.3	0.34	0.28	0.16	5	5	3.5	2.7	18.0
	14	Wet Tuff	0.5	1.3	0.10	0.06	0.11	5	5	3.2	2.0	16.0
	15	Dry Tuff	0.5	1.3	0.045	0.03	0.06	5	5	3.2	1.7	18.0
	16	Oil Shale	0.5	1.0	0.28	0.11	0.11	5	5	3.2	2.3	16.0
	17	Dolomite	0.5	0.6	0.85	0.30	0.10	5	5	2.5	2.8	14.0
	18	Limestone	0.5	0.6	0.4	0.67	0.10	5	5	2.5	2.7	14.0
	19	Halite	0.5	0.6	0.25	0.30	0.05	5	5	2.0	2.2	15.0

1. Tillotson, J. H., "Metallic Equations of State for Hypervelocity Impact," General Atomic Report GA-3216, July 1962.
2. Walsh, J. M., unpublished notes.
3. Walsh, J. M., J. H. Tillotson, and W. E. Johnson, "Theory of Hypervelocity Impact, Quarterly Progress Report," General Atomic Report GACD-4518, July 1963.
4. Evans, M. W., and F. H. Harlow, "The Particle-in-Cell Method for Hydrodynamics Calculations," Los Alamos Scientific Laboratory Report LA-2139, November 1957.
5. For these two materials the constants are valid only for pressures less than 150 kbar. For a formulation of the high pressure equation of state of these materials, see Allen, R. T., "Equation of State of Rocks and Minerals," General Atomic Report GAMD-7834, March 1967.

2.3.2 Elastic-Plastic Constitutive Relation

Section 2.2.1 discussed the modification of the material velocity and internal energy fields due to the effects of the stress deviators in subroutine SPHASE. This section will discuss the elastic-plastic material model incorporated in the HELP code to calculate the stress deviators.

Subroutine SPHASE has two major tasks: first, it computes instantaneous strain rate deviators and updates the stress deviators, and second, it updates the cell momenta and energy to account for the effects of the stress deviators (as discussed in Section 2.2.1). This section will discuss the first task of SPHASE: the computation of the strain rate deviators and the updating of the stress deviators, including a description of the von Mises yield condition.

2.3.2.1 Strain Rate Deviators

As a first step in computing stress deviators, the instantaneous strain rate deviators are computed from the velocity field, i.e., for cylindrical coordinates

$$\dot{\varepsilon}_{zz} = v_y - \frac{1}{3}\left(u_x + v_y + \frac{u}{x}\right)$$

$$\dot{\varepsilon}_{rr} = u_x - \frac{1}{3}\left(u_x + v_y + \frac{u}{x}\right)$$

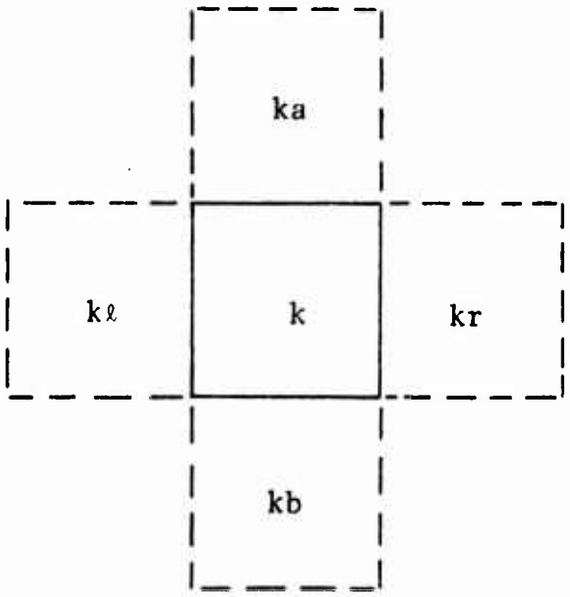
$$\dot{\varepsilon}_{rz} = (u_y + v_x)/2 \quad ,$$

where the x and y subscripts denote partial differentiation with respect to those variables. Here the necessary space derivatives of the velocity field are computed (centered at cell centers) in a straightforward manner, i.e.,

$$v_y = \frac{v(ka) - v(kb)}{2\Delta y}$$

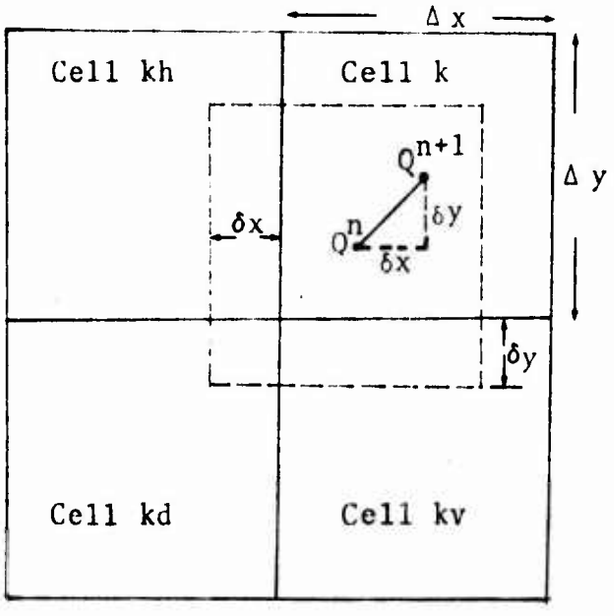
$$v_x = \frac{v(kr) - v(k\ell)}{2\Delta x}$$

etc.



The above calculation gives only the strain rate deviator for that material which is at cell center at the beginning of the time step, and it is necessary to recognize that constitutive equations are to be applied only along a particle path. To this end, one must take a convective derivative. Consider the time n position Q^n of a particle which, at time n+1, will be at a cell center. This point will be offset from the cell center by δx and δy where

$$\delta x = -u(k)\ \Delta t$$
$$\delta y = -v(k)\ \Delta t \quad .$$



Now we wish to determine, by interpolation using the value of the strain rate deviators at cell centers, the strain rate deviators at the offset point Q^n. This is done by weighting the contribution of neighboring cells in proportion to their overlap areas with a rectangle of cell dimensions, centered at Q^n. For example, referring to the above sketch, the weighting factors are

for cell k

$$WK = (\Delta x - |\delta x|)\ (\Delta y - |\delta y|)$$

for cell in horizontal direction

$$WKH = |\delta x|\ (\Delta y - |\delta y|)$$

for cell in vertical direction

$$WKV = |\delta y|\ (\Delta x - |\delta x|)$$

for cell in diagonal direction

$$WKD = |\delta x|\ |\delta y|$$

so that the resulting interpolated value of a strain rate deviator component, say $\dot{\varepsilon}'_{rz}$, is given by

$$\dot{\varepsilon}'_{rz} = \frac{(WK)\ \dot{\varepsilon}_{rz}(k) + (WKH)\dot{\varepsilon}_{rz}(kh) + (WKV)\dot{\varepsilon}_{rz}(kv) + (WKD)\dot{\varepsilon}_{rz}(kd)}{WK + WKH + WKV + WKD} \quad .$$

Having thus determined $\dot{\varepsilon}'_{zz}$, $\dot{\varepsilon}'_{rr}$, $\dot{\varepsilon}'_{rz}$ for the mass which will be at the center of cell k at the end of the time step, we can now calculate the strain deviator increments. For example,

$$\Delta\varepsilon_{rz} = \dot{\varepsilon}'_{rz}\ \Delta t \quad .$$

2.3.2.2 Stress Deviators

These strain deviator increments, $\Delta\varepsilon_{ij}$, are next used in the material constitutive equation to update the stress deviator tensor, s_{ij}. To do this one needs the time n stress deviators for the particle in question and these s'_{ij} are determined (from stored cell centered values of the s_{ij} at time n) by an identical area weighting procedure as described above for strain rate deviators.

For an elastic-plastic material the following calculation then (provisionally) updates the s_{ij} according to

$$s_{ij}^{n+1} = s'_{ij} + 2G\,\Delta\varepsilon_{ij}$$

where G is the material rigidity modulus; i.e., the three components of interest are given by

$$s_{zz}^{n+1} = s'_{zz} + 2G\,\Delta\varepsilon_{zz}$$

$$s_{rr}^{n+1} = s'_{rr} + 2G\,\Delta\varepsilon_{rr}$$

$$s_{rz}^{n+1} = s'_{rz} + 2G\,\Delta\varepsilon_{rz} \quad .$$

The provision (above) is that the yield condition for the material not be violated.

2.3.2.3 Yield Criterion

HELP employs the von Mises yield condition which requires that

$$s_{ij}s_{ij} \leq 2Y^2$$

where $s_{ij}s_{ij}$ is twice the second invariant of the stress tensor; i.e.,

$$s_{ij}\, s_{ij} = s_{rr}^2 + s_{zz}^2 + s_{\theta\theta}^2 + 2s_{rz}^2 \quad ,$$

where $s_{\theta\theta} = -(s_{rr} + s_{zz})$ is the hoop stress and Y is the shear yield strength. If the yield condition is violated during a time step, the deviatoric stress components are proportionately reduced normal to the yield surface. If the yield condition is not violated, the deformation is purely elastic and the stresses are not reduced.

A variable yield strength

$$Y = (Y_o + Y_1\, \mu + Y_2\, \mu^2)\ (1 - E_I/E_M)$$

is defined in the code (in subroutine STRNG) to account for the increase in strength at high compressions and the decrease in strength at elevated energies. In this equation Y_o, Y_1 and Y_2 are input constants, $\mu = \rho/\rho_o - 1$ where ρ is the density and ρ_o the reference density, E_I is the specific internal energy and E_M is the melt energy. This expression is never allowed to be negative, however, so that once the melting point energy is exceeded the material yield strength is set to zero. It should be noted that cells (including multi-material cells) containing material which has failed in tension (see the following subsection) are assumed to have no shear yield strength. Further modifications to the yield criterion for multimaterial cells are discussed in Section 4.6.2.

2.3.3 Material Failure in Tension

The tensile failure criterion for pure cells is a simple one based on relative volume. When the relative volume ($V_o/V = \rho/\rho_o$) of material in a pure cell reaches a value which is greater than a specified maximum dilitation (an input constant for each material, $AMDM_i$), the material in the cell is considered to have failed. A cell which has failed is not allowed to sustain tensile hydrostatic pressures nor, as noted in the preceding subsection, to have shear yield strength.

The tensile failure criterion is more stringent for multi-material cells in that they are not allowed to sustain negative pressures, regardless of the relative volumes of the materials they contain, since, presumably, the negative pressure would cause the materials to separate at the interface, thus immediately relieving the tension.

2.3.4 Detonation of High Explosives

To model high explosives a detonation procedure has been incorporated into HELP as well as special prescriptions which maintain a sharp boundary between the detonated and undetonated explosive.

2.3.4.1 Detonation Procedure

The detonation procedure incorporated into HELP makes no assumptions about the geometric shape of the explosive package, the number of explosives in the problem, or the number of initiation points, although certain variable dimensions allow a maximum of ten primary and ten secondary initiation points.

Primary initiation points may be located in any configuration within an explosive package, and may have a time delay associated with them. A primary initiation point can be detonated at any time on or after time zero (the start of the calculation). For example, a string of five detonators may be initiated sequentially or simultaneously, depending upon the desired results.

Secondary initiation points which are used to detonate around obstacles may also be located in any configuration throughout the high explosive packages. The location of secondary initiation points should be selected carefully if they are to perform the intended function, that of detonating around obstacles. The initiation times for each of the secondary initiation points are computed by the same method used to compute the detonation time for a cell.

Briefly, the numerical method for handling the detonation of a high explosive is to calculate the detonation time of each explosive cell in the grid based upon the cell's location and the constant detonation velocities of each high explosive through which the detonation front passes between the primary sources and the cell to be detonated.

The high explosive detonation routine employed in the HELP code allows multi-initiation of multiple, adjacent, high explosive packages. For the case of adjacent HE packages the current method is restricted to explosives whose detonation velocities and C-J pressures are within ten percent of one another so that the degree to which one of the detonations is overdriven is minimal and can be neglected.

Figure 2.4 illustrates the method employed to compute the detonation times of cells in adjacent high explosive packages. In this example the primary initiation point, P_o, is in the HE_1 package, and cell K1 is a pure cell in the HE_1

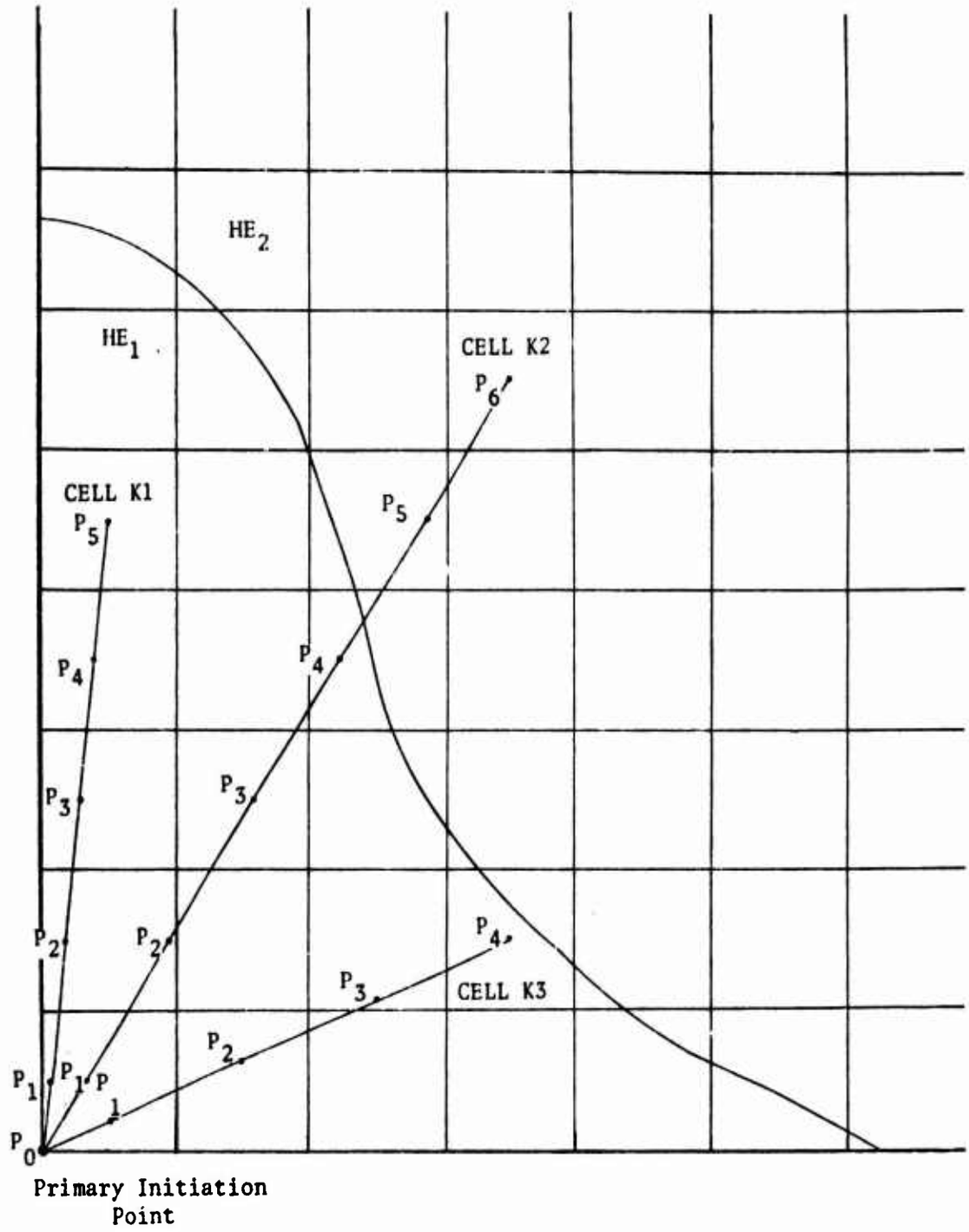


Figure 2.4--Illustration of the method employed to compute detonation times of HE cells.

package, cell K2 is a pure cell in the HE_2 package, and cell K3 is an interface cell containing both HE_1 and HE_2. In all cases the time to detonate a cell is the sum of the time increments, t_i, associated with the time for the detonation front to travel from the points P_{i-1} to P_i, which lie on the line from the primary initiation point to the center of the cell. If this line is at an angle which is less than or equal to 45^o with respect to the x-axis, the P_i are the intercepts of that line with the centers of the grid columns (e.g., the line to cell K3). If the angle is greater than 45^o, the P_i are the intercepts of that line with the centers of the grid rows (e.g., the lines to cells K1 and K2). If both points, P_{i-1} and P_i, are in pure HE_1 cells, the detonation time, t_i, is based on the HE_1 detonation velocity, D_1. If the second point P_i, is in an interface cell containing both HE_1 and HE_2 explosives, the detonation time, t_i, is based on the maximum of the HE_1 and HE_2 detonation velocities: $(D_1, D_2)_{max}$.

Referring to Figure 2.4, all of the time increments for cell K1 are computed using D_1. For cell K2, however, the first three increments are based on D_1, the next two on $(D_1,D_2)_{max}$, and the last one on D_2. For cell K3, all but the last are based on D_1 and the last is a function of $(D_1,D_2)_{max}$.

The HELP code has been applied to the jet formation process associated with lined shaped charges in which the detonation wave has been "shaped" by the insertion of inert material into the high explosive package.[12] In order to accomplish this it was necessary to incorporate the capability of detonating around obstacles into the code. Figure 2.5 shows a disc of inert material situated in a package of HE in which detonation is initiated at a point on the axis of symmetry. The primary initiation point is shown as Point 1. The detonation time of each cell contained in

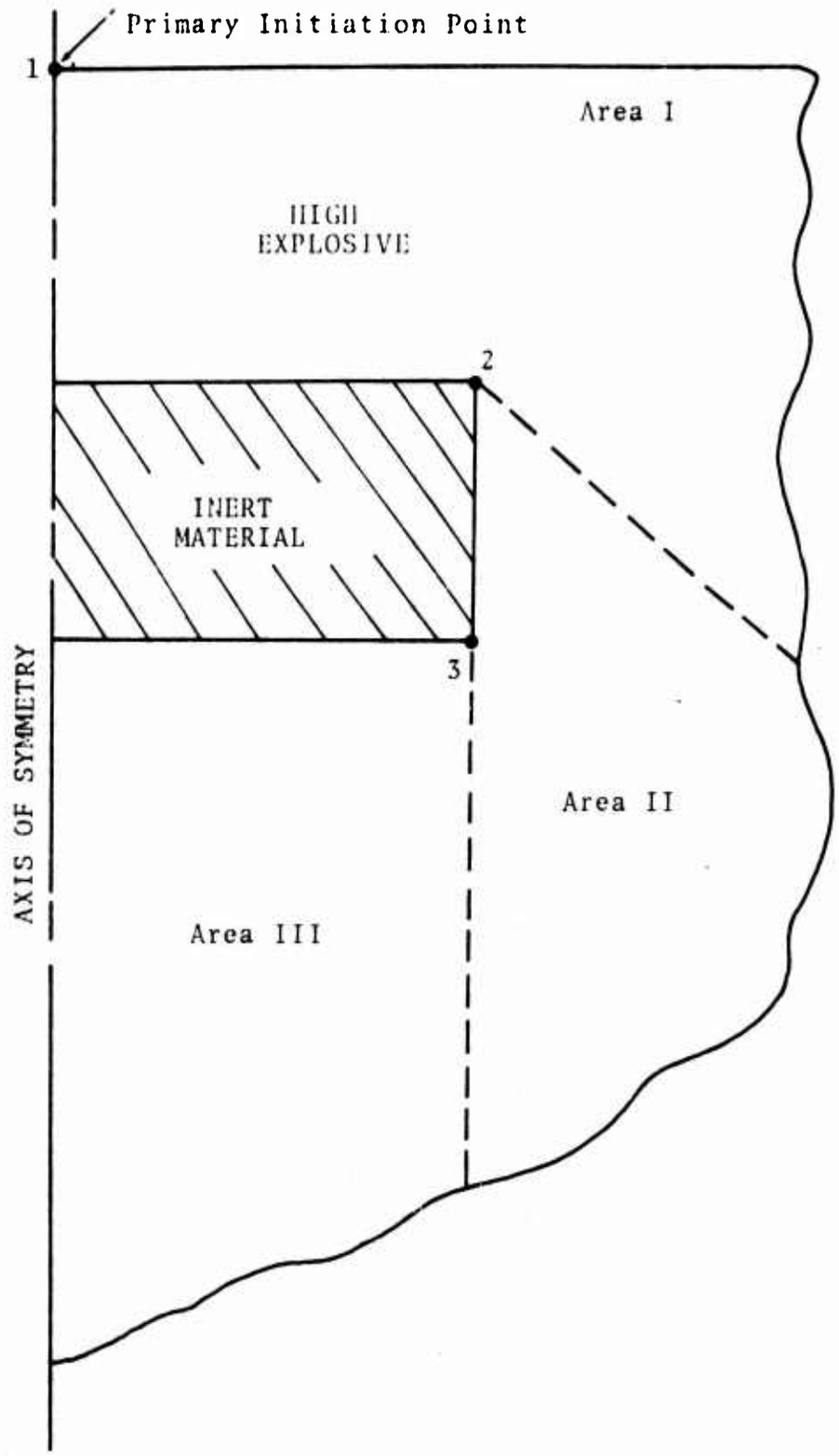


Figure 2.5--Detonation around obstacles.

area I is based on its distance from Point 1 and the detonation velocity of the HE. Similarly, the cells in areas II and III will be detonated at times based on their distances from the secondary initiation points, 2 and 3, and the detonation velocity. Points 2 and 3 in Figure 2.5 act as secondary initiation points for their respective areas as soon as the HE in the cells containing these points has been detonated.

Both the primary and secondary detonation points, as well as their corresponding detonation regions, are part of the initial conditions and are specified by the input cards.

At the beginning of the problem the time to detonate an explosive cell is computed and stored in the DETIM array (see Section 7.2.7). On the cycle for which the value of T in the calculation equals or exceeds the detonation time associated with a given cell, the chemical energy of detonation of the explosive (E_o in Table 2.1) is added to the cell. If the cell contains more than one explosive, the appropriate chemical energy of detonation of each high explosive is added to the cell.

Each cycle, ADDENG prints the row (I) and column (J) of every cell detonated on that cycle. The total energy released on that cycle is also printed, and the theoretical energy of the grid (ETH) is updated accordingly.

2.3.4.2 Definition of the Detonation Front

Because the detonation procedure used by HELP detonates all of the explosive in a cell at one time, the detonation front is approximated by the boundaries between detonated and undetonated cells. In order to prevent small numerical signals from propagating through the grid ahead

of the detonation front, it is necessary to place certain restrictions on cells containing undetonated explosive during the three phases of the code.

There are two restrictions necessary to prevent the pressure effects from propagating undesired numerical signals. The first of these is implemented in CDT and requires that any cell containing undetonated explosive have zero pressure. For these cells, the equation of state routine, EQST, is not called by CDT. This restriction is necessary since the explosive equation of state applies only to the detonated explosive. The second restriction, implemented in the pressure effects subroutine, HPHASE, requires the pressure and velocities to be zero at the four boundaries of an interface or pure cell containing undetonated explosive, even if one or more of the neighbor cells contains detonated explosive.

In the strength effects phase, SPHASE, any mixed or pure cell which contains explosive is assumed to have no strength and its deviator stresses are set to zero. This assumption applies to cells containing detonated as well as undetonated explosive.

In subroutine TPHASE, the transport phase of the code, no mass, momenta, or energy is transported into or out of a cell containing undetonated explosive. These transport restrictions are coded into DMCALC as well, which computes the mass of each material to be transported across the four boundaries of an interface cell. If an interface cell contains undetonated explosive, no mass of any material is transported into or out of the cell.

CHAPTER III

GRID BOUNDARY CONDITIONS

3.1 BASIC ASSUMPTIONS

Additional specifications are required for cells which border the grid boundaries because necessary quantities are not defined for neighbor cells which would be outside the grid. The code provides for a transmittive or reflective boundary condition at the bottom and a reflective boundary condition at the left (see Figure 3.1). Boundary conditions for grid boundary cells are then derived by assuming fictitious neighbor cells outside the grid. For transmittive boundaries the flow variables are the same in the fictional cell as in the border cell, and for reflective boundaries the state is assumed to be the same in the fictional cell except that the velocity component normal to the boundary has the opposite sign.

These assumptions, as they apply to subroutines [illegible]SE, HPHASE, and TPHASE, are described in more detail in the sections that follow. Special prescriptions for moving tracer particles on and near the grid boundaries as well as special edited quantities which indicate the work done and the mass, momentum, and energy fluxes at the grid boundaries are described as well.

3.2 STRENGTH PHASE (SPHASE)

The SPHASE grid boundary conditions will be discussed as they pertain to the three subtasks of SPHASE:

1. the calculation of cell centered strain rates;
2. the interpolation of cell centered stresses and strain rates to account for convection;
3. the application of deviator stresses to update cell velocities and internal energies.

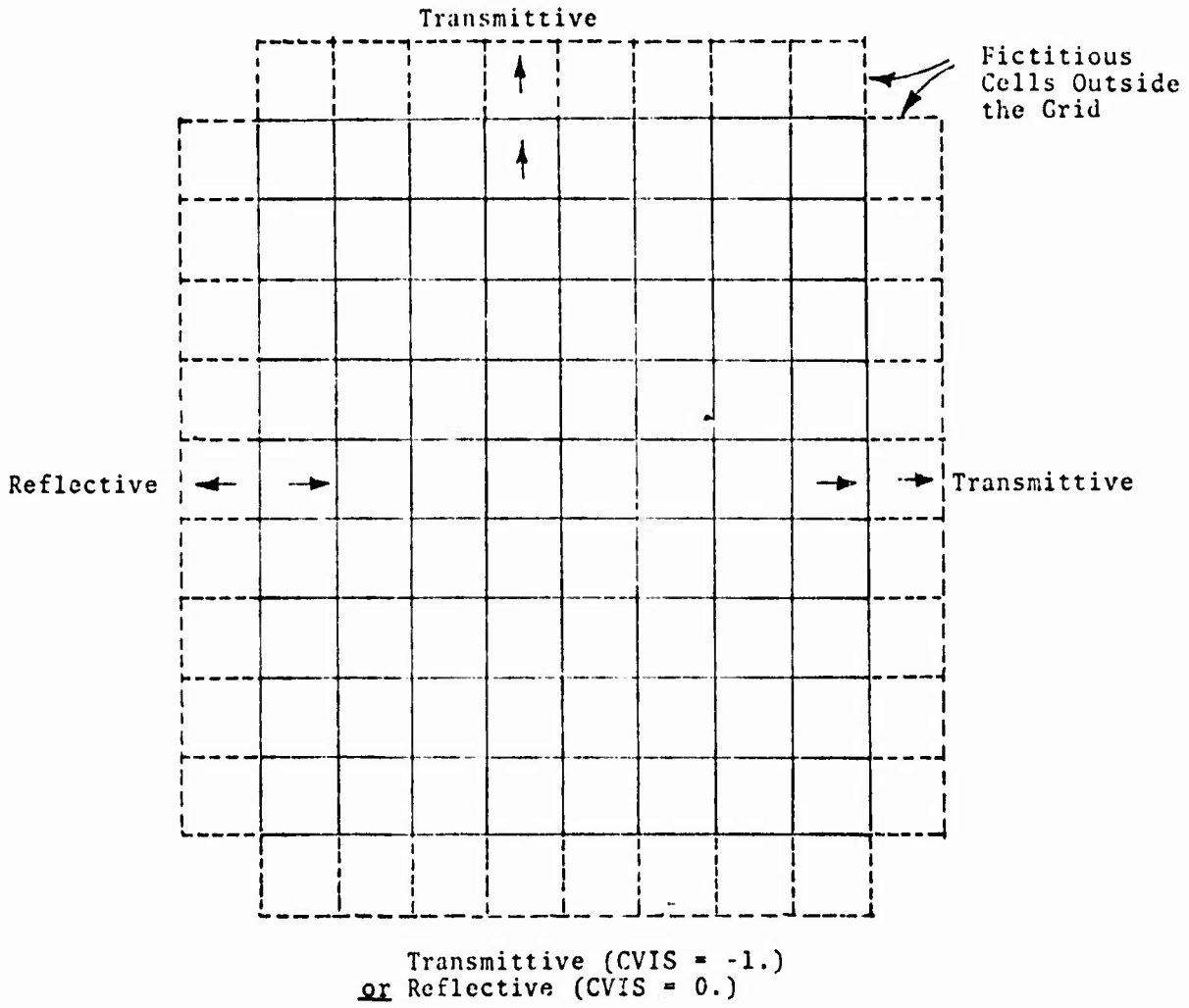


Figure 3.1--The HELP grid boundary conditions.

Within each subtask the transmittive and reflective boundary cases will be discussed.

3.2.1 Definition of Strain Rate Derivatives for Cells at a Grid Boundary

The strain rates are computed using space derivatives centered at cell centers. For example the finite difference analogs of $\partial v/\partial y$ and $\partial u/\partial y$ for cell k are

$$\frac{\partial v}{\partial y} = \frac{v(ka) - v(kb)}{.5\Delta y_{j-1} + \Delta y_j + .5\Delta y_{j+1}}$$

$$\frac{\partial u}{\partial y} = \frac{u(ka) - u(kb)}{.5\Delta y_{j-1} + \Delta y_j + .5\Delta y_{j+1}} \quad ,$$

with ka and kb being the indices of the cells above and below cell k, respectively, and Δy being the axial dimension of the corresponding cell. However, if cell k is at the top transmittive grid boundary, these space derivatives are approximated by

$$\frac{\partial v}{\partial y} = \frac{v(k) - v(kb)}{.5\Delta y_{j-1} + 1.5\Delta y_j}$$

$$\frac{\partial u}{\partial y} = \frac{u(k) - u(kb)}{.5\Delta y_{j-1} + 1.5\Delta y_j} \quad ,$$

which assumes that the fictitious cell above cell k has the same velocity components and axial dimension as cell k. If, on the other hand, cell k is at the bottom grid boundary and that boundary is reflective, these space derivatives become

$$\frac{\partial v}{\partial y} = \frac{v(ka) - (-v(k))}{1.5\Delta y_j + .5\Delta y_{j+1}}$$

$$\frac{\partial u}{\partial y} = \frac{u(ka) - u(k)}{1.5\Delta y_j + .5\Delta y_{j+1}} \quad ,$$

which assumes that a fictitious cell below the grid has the same velocity components and axial dimension as cell k, except that the velocity component normal to that boundary has the opposite sign.

Similar approximations are made for $\partial u/\partial x$ and $\partial v/\partial x$ for cells at the right transmittive grid boundary and for cells at the left reflective grid boundary.

3.2.2 Definition of Interpolated Strain Rates and Stresses for Cells at a Grid Boundary

SPHASE computes the deviator stresses of a particle that will be at the center of cell k at the end of the time step. The strain rates and time n stresses of such a particle are computed by taking an area weighted average of cell centered strain rates and stresses. (See Section 2.3.2.1.) If cell k is at a grid boundary, and the velocity field is such that the material is flowing from outside the grid, then it is assumed that the fictitious cells outside the grid have the same stresses and strain rates as the contiguous cells inside the grid.

3.2.3 Definition of Velocities and Deviator Stresses at Grid Boundaries

The cell velocities and internal energies are updated in SPHASE using cell boundary stresses and velocities which

are averages of the cell centered quantities. For example, the stresses and velocities at the right boundary of cell k, which is not at a grid boundary, are

$$\bar{u}^r = \frac{u(r) + u(kr)}{2}$$

$$\bar{v}^r = \frac{v(k) + v(kr)}{2}$$

$$\bar{s}^r_{rr} = \frac{s_{rr}(k) + s_{rr}(kr)}{2}$$

$$\bar{s}^r_{rz} = \frac{s_{rz}(k) + s_{rz}(kr)}{2} \quad .$$

If, however, cell k is at the right transmittive grid boundary then the stresses and velocities of the fictitious cell outside the grid are assumed to be the same as those of cell k, and the boundary stresses and velocities are

$$\bar{u}^r = u(k)$$

$$\bar{v}^r = v(k)$$

$$\bar{s}^r_{rr} = s_{rr}(k)$$

$$\bar{s}^r_{rz} = s_{rz}(k) \quad .$$

And if cell k is at the left reflective grid boundary, the following equations define the stresses and velocities at the left boundary:

$$\bar{u}^\ell = 0$$

$$\bar{v}^\ell = v(k)$$

$$\bar{s}^\ell_{rz} = 0$$

$$\bar{s}^\ell_{rr} = s_{rr}(k) \quad ;$$

i.e., the shear stress and the velocity component normal to the boundary are set to zero, and the normal stress and tangential velocity component are assumed to be equal to the cell k values. Likewise, the stresses and velocities at the bottom grid boundary, when it is reflective, are

$$\bar{u}^b = u(k)$$

$$\bar{v}^b = 0$$

$$\bar{s}^b_{rz} = 0$$

$$\bar{s}^b_{zz} = s_{zz}(k) \quad .$$

3.2.4 Correction to the Theoretical Energy for Work Done at Grid Boundaries in SPHASE

The theoretical energy of the grid, ETH, is updated only for the work done at the transmittive grid boundaries. At the right boundary

$$\delta E_r = [\, s_{rr}(k) \cdot u(k) + s_{rz}(k) \cdot v(k) \,] \cdot A_r \cdot \Delta t \quad ,$$

where s_{rr}, s_{rz}, u, v, A_r are the deviator stress normal to the right boundary, the shear stress, the velocity component normal to the right boundary, the velocity component tangent to the right boundary, and the area of the right cell face, respectively. Similarly, at the top transmittive grid boundary

$$\delta E_t = [\, s_{zz}(k) \cdot v(k) + s_{rz}(k) \cdot u(k) \,] \cdot A_t \cdot \Delta t \quad ,$$

and at the bottom grid boundary, when it is transmittive (CVIS = - 1),

$$\delta E_b = [s_{zz}(k) \cdot v(k) + s_{rz}(k) \cdot u(k)] \cdot A_b \cdot \Delta t \quad .$$

When the bottom boundary is reflective, $\delta E_b = 0$, since in that case the normal velocity, v, and the shear stress, s_{rz}, are zero.

3.3 HYDRODYNAMIC PHASE (HPHASE)

The effects of pressure are computed by using pressures and velocities defined at cell boundaries. If a cell boundary is part of the grid boundary, the definition of the pressure and velocity at that boundary depends on whether or not that boundary is reflective or transmittive.

3.3.1 Definition of Velocities and Pressures at Transmittive Grid Boundaries

If the grid boundary is transmittive, the cell boundary pressure is assumed to be equal to the cell centered pressure. Likewise, the velocity component normal to the boundary is assumed to be equal to the cell centered velocity component. For example, if cell k is at the top transmittive boundary

$$\bar{p}^a = P(k)$$

$$\bar{v}^a = v(k) \quad ,$$

and if cell k is at the right transmittive boundary

$$\bar{p}^r = P(k)$$

$$\bar{u}^r = u(k) \quad .$$

Because the transmittive boundary condition assumes all material outside the grid is at the same state as the material

in the border cells of the grid, the user should, by rezoning, prevent a strong shock from moving across a transmittive boundary. In such an instance a numerical signal can be generated by the boundary condition, and the stresses in the material behind the shock can become noisy and meaningless.

3.3.2 Definition of Velocities and Pressures at Reflective Grid Boundaries

If the grid boundary is reflective, the cell boundary pressure is assumed to be equal to the cell centered pressure. However, the velocity component normal to the boundary is assumed to be zero.

For example, if cell k is at the left reflective grid boundary (an axis of symmetry in cylindrical coordinates)

$$\bar{p}^{\ell} = P(k)$$

$$\bar{u}^{\ell} = 0 \quad ,$$

and if cell k is at the bottom grid boundary which is acting as a reflective boundary (CVIS = 0),

$$\bar{p}^{b} = P(k)$$

$$\bar{v}^{b} = 0.$$

3.3.3 Correction to Theoretical Energy for Work Done at Grid Boundaries in HPHASE

The theoretical energy of the grid, ETH, is updated in HPHASE for the work done at the transmittive grid boundaries. At the top transmittive boundary

$$\delta E_t = -P(k) \cdot v(k) \cdot A_t \cdot \Delta t \quad ,$$

and at the right transmittive boundary

$$\delta E_r = -P(k) \cdot u(k) \cdot A_r \cdot \Delta t \quad ,$$

where A_t and A_r are the areas of the top and right cell faces, respectively. If the bottom boundary is transmittive

$$\delta E_b = P(k) \cdot v(k) \cdot A_b \cdot \Delta t \quad ,$$

and if it is reflective, $\delta E_b = 0$, since $\bar{v}^b = 0$.

3.4 TRANSPORT PHASE (TPHASE)

Material is transported through the grid by computing mass, momentum and energy fluxes at each of the four boundaries of every cell. If one of these boundaries is a grid boundary, these transport terms are determined by the following conventions for transmittive and reflective boundaries, and the theoretical energy is adjusted accordingly.

3.4.1 Transport of Mass, Momentum and Energy Across Transmittive Grid Boundaries

The velocity of the material at and beyond a grid boundary is assumed to be the same as that of the boundary cell. The transport velocity used to compute the mass transported across a grid boundary is therefore the cell centered velocity. However, if the cell centered velocity component normal to the grid boundary is such that the material is moving into the grid, the mass transport term at that boundary is set to zero. For example, if cell k is at the top transmittive grid boundary and $v(k) > 0$, then

$$\delta m_a = v(k) \cdot \Delta t \cdot A_t \cdot \rho(k) \quad .$$

On the other hand, if $v(k) \leq 0$ then $\delta m_a = 0$. Likewise, if cell k is at the right transmittive boundary and $u(k) > 0$, then

$$\delta m_r = u(k) \cdot \Delta t \cdot A_r \cdot \rho(k) \quad .$$

If $u(k) \leq 0$, however, $\delta m_r = 0$.

For the case where $\delta m > 0$, the momentum and energy transported across grid boundaries are determined by the donor cell method, i.e.,

$$\delta(mu) = \delta m \cdot u(k)$$

$$\delta(mE) = \delta m \cdot E(k) \quad .$$

The code does not have provisions for feeding material into the grid across any of the three transmittive grid boundaries; however, TPHASE can be easily modified by the knowledgeable user to do so.

3.4.2 Change of Momentum for Cells at Reflective Grid Boundaries in TPHASE

Even though no mass is transported across a reflective boundary, the momentum of a cell at a reflective boundary is modified in TPHASE.

A useful way to think about reflective boundaries, such as the axis or grid boundaries, is to take them to be planes (or a line) of symmetry through which equal and opposite transports occur in both directions.

Assume, for example, that the mass transport calculation at the axis indicates that a mass δm^* would pass through the axis (i.e., $u < 0$). Then an equal and opposite transport is assumed from the left. The net effect on cell mass is zero.

However, there is an effect on the cell momentum. At the axis, momentum δmu is lost and a momentum $\delta m(-u)$ is gained. The net result is a momentum change of $-2\delta mu$. Thus the momentum equation has to have this correction term; specifically

$$\Delta_T mu = \sum_i \delta(m_i u_i) - 2\delta mu \quad ,$$

where the subscript i denotes the four sides of the cell. Similarly, it can be shown that the correction term at the bottom grid boundary, when it is reflective, is $-2\delta mv$.

In the case of multimaterial cells, the energy equation must also be modified. For example, consider a cell on the axis of symmetry, where u is its pre-TPHASE radial velocity and u' is its post-TPHASE radial velocity. The correction to the cell's thermalized kinetic energy (see Section 4.6.3) is as follows:

$$-\frac{\delta m}{2}(u-u')^2 + \frac{\delta m}{2}(-u-u')^2 = 2\delta muu'$$

where the first term corresponds to the mass that "left" with velocity u, and the second term corresponds to the mass that "entered" with velocity -u. Similarly, the correction term for a multimaterial cell at the bottom grid boundary, when it is reflective, is $2\delta mvv'$.

---

* The discussion above is independent of how one calculates δm. However, in HELP, at the axis $\delta m = \rho u(\pi \Delta r \Delta z)\Delta t$ if $u < 0$, and $\delta m = 0$ if $u > 0$. The area term, $\pi \Delta r \Delta z$, is the area of the surface which cuts through the center of the cell.

3.4.3 Correction to Theoretical Energy for Energy Transported Across Grid Boundaries in TPHASE

In TPHASE the theoretical energy of the grid, ETH, is adjusted to account for the total energy of the mass that leaves the grid across the transmittive grid boundaries. For example, if cell k is at the top transmittive grid boundary, and $\delta m_t > 0$, then

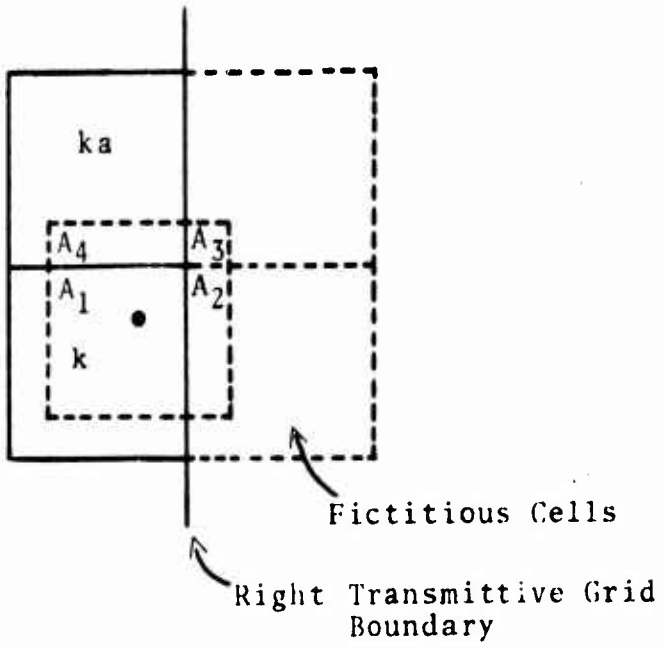
$$\delta E_t = \delta m_t \cdot \left(E(k) + 1/2\left(u(k)^2 + v(k)^2\right)\right)$$

where E(k) is the specific internal energy of cell k. There is no mass loss and therefore no energy loss at a reflective boundary.

3.5 TRACER PARTICLE MOTION NEAR GRID BOUNDARIES

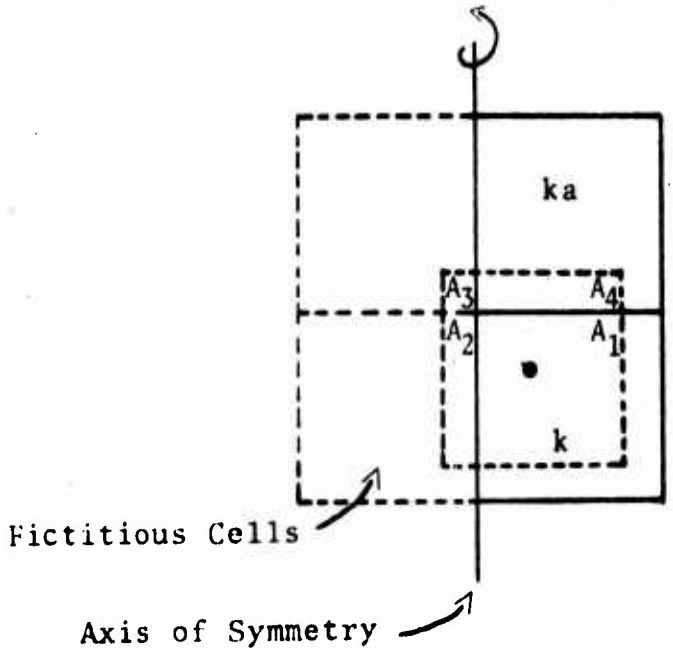
As elsewhere in HELP, subroutine MOVTCR assumes the velocity of material outside a transmittive grid boundary is equal to that of the material in the cells bordering the grid boundary. Therefore, if the overlap area (described in Section 4.1.2) used to compute the velocity of a given particle extends into a fictitious cell outside the grid, the velocity of that fictitious cell is assumed to be equal to its neighbor inside the grid. For example, as illustrated in the following figure, the radial velocity of the particle is defined as follows:

$$\bar{u} = \frac{A_1 u(k) + A_2 u(k) + A_3 u(ka) + A_4 u(ka)}{A_1 + A_2 + A_3 + A_4} .$$



If the grid boundary is reflective, however, the velocity component normal to that boundary in the fictitious cell is assumed to have the opposite sign. If the particle is near the axis of symmetry, as illustrated in the following figure, the radial velocity of the particle is defined as follows:

$$\bar{u} = \frac{A_1 u(k) + A_2 \left(-u(k)\right) + A_3 \left(-u(ka)\right) + A_4 u(ka)}{A_1 + A_2 + A_3 + A_4} \quad .$$



An analogous procedure is followed for computing the axial component of velocity for a particle near the bottom grid boundary when it is reflective (CVIS = 0).

The coordinates of the tracer particles are stored in grid units. If the new position of a particle is computed to be beyond the grid, its x-coordinate and/or y-coordinate is set equal to the grid boundary line it has exceeded, i.e., 0, IMAX, or JMAX, since it is not possible to track a particle outside the grid. Furthermore, once a particle is moved onto a grid boundary, it is not moved off of that boundary, only along it.

3.6 EDITING OF FLUXES AT GRID BOUNDARIES

In SPHASE and HPHASE the total work done at each of the transmittive grid boundaries is accumulated, as well as the internal energy lost by each material package due to work done at these boundaries (calculated in subroutine EOUT). The total work is printed by subroutine EDIT under the heading WORK DONE and is included in the total printed under the heading IE OUT.

In TPHASE, the mass that leaves the grid and its associated total energy and momenta are accumulated and are printed by subroutine EDIT under the headings MASS OUT, ENERGY OUT, MU OUT, and MV OUT, respectively. The kinetic and internal energy lost by each material package due to transport across grid boundaries is also accumulated and is included in the totals printed by EDIT under the headings KE OUT and IE OUT.

CHAPTER IV

MATERIAL INTERFACES AND INTERFACE CELLS

The following sections describe the HELP method of defining and propagating material interfaces through an Eulerian grid and discuss the special treatment given to multimaterial cells. The use of tracer particles to define the interfaces and the manner in which these particles are moved are discussed in Section 4.1. The method for storing information on several materials in an interface cell and for converting a pure cell into an interface cell is described in Section 4.2. The procedures for calculating fractional cell face areas (used in the transport of materials across interface cell boundaries) are given in Section 4.3. The adjustment of mass transport terms to exactly evacuate a material when an interface leaves a cell and to prevent overemptying of a material is discussed in Section 4.4. In Section 4.5 the characteristics of the three types of interface cells - multimaterial, free surface and slipline - are given. Finally, the special treatment necessary to define various cell and material properties for multimaterial cells is given in Section 4.6.

4.1 DEFINITION OF MATERIAL INTERFACES

4.1.1 Use of Tracer Particles

A set of massless tracer particles are initially positioned along the boundary of each material package. These tracers are numbered such that the package is on the left as one proceeds between any two consecutive tracers. The material interface therefore is defined by the line segments connecting these tracers. The initial spacing of the tracers is specified when they are generated by TSETUP and can be maintained in a specified region as the calculation progresses by periodic calls to ADDTCR. (See description of ADDTCR in Section 8.2.)

To prevent the boundaries of adjacent packages from crossing, the tracers along the common boundary coincide exactly. Because these identical points are moved with exactly the same velocity, they remain superimposed during the entire calculation.

To allow for spatially disconnected subpackages, the code employs the following convention. The last point of a material package or subpackage is a dummy point, its x-coordinate is -1000, its y-coordinate is 0. A given material package can be divided into any number of spatially disconnected subpackages.

4.1.2 Movement of Tracer Particles

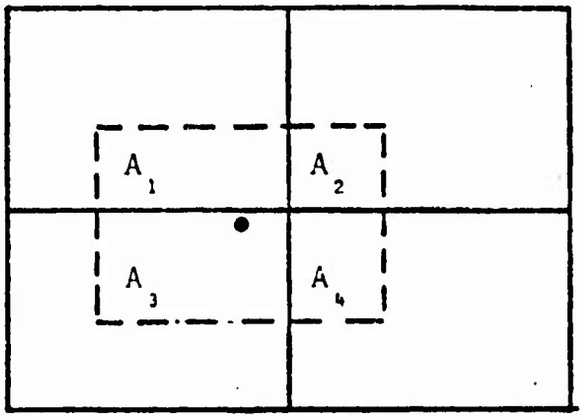
Each material package is circumscribed by a series of massless tracer particles, which are propagated each time step a distance*

$$\Delta\vec{x}_i = \vec{u}_i \, \Delta t_{INFACE} \tag{4.1}$$

where $\vec{u}_i$ is a local, density weighted, average velocity vector for the continuum, determined by an area overlap method which gives weight to velocities in the surrounding cells. Specifically, a rectangle of cell dimensions is superimposed on the particle to be moved and then

$$\vec{u}_i = \frac{\sum_L \vec{w}(L) \; A_L \rho_L}{\sum_L A_L \rho_L} \quad , \tag{4.2}$$

*Since the tracer coordinates are in grid line units, this distance is converted from centimeters to cell units.



where $\vec{w}(L)$ is the cell-centered velocity* of the overlapped cell L, A_L is the overlap area and ρ_L is the total cell mass divided by the total cell volume. This procedure gives a spatially continuous velocity field for particle propagation.

The density weighting causes the particles to follow the motion of the denser material and prevents the interface from becoming unstable in regions where there are large density discontinuities.

Infrequently, the sum of the area terms, A_i, equals zero, which indicates all of the overlap cells associated with a given tracer particle are empty and it cannot be moved in the usual manner. When this occurs, the code will move the isolated tracer to the position of one of its immediate neighbors. However, if the isolated tracer is on a grid boundary, it will not be moved off of that boundary, only along it.

If INFACE is subcycled (CYCMX > 1), then the time step, Δt_{INFACE}, used to move the tracers on a given subcycle, is the time step computed by CDT divided by the number of INFACE subcycles, i.e.

$$\Delta t_{INFACE} = \frac{\Delta t_{CDT}}{CYCMX} \quad . \qquad (4.3)$$

*These cell-centered velocities have been updated by HPHASE (pressure effects) and SPHASE (strength effects) for this time step.

The passive tracers (XP, YP), used only for tracking the mass within the package boundaries, are moved by the same method, except that if INFACE is subcycled they are moved only on the last subcycle using the Δt_{CDT} time step.

4.2 CREATION OF INTERFACE CELLS

In the discussion that follows an interface cell is any cell whose boundary is intersected by a material interface, whereas a pure cell is completely within one of the material package boundaries. An interface cell may contain one or more distinct materials, it may contain a void and therefore be a free surface cell, and it may contain a slipline. The characteristics of the various types of interface cells are given in Section 4.5. The way in which interface cells are flagged and the additional information stored for each of them are discussed in the sections below.

4.2.1 Use of MFLAG Array

In all cases, the value of MFLAG(k) of an interface cell k is greater than 100. The value of MFLAG(k) of a pure cell k, however, is less than or equal to the number of material packages in the grid (0, 1, ..., NMAT).

For interface cells, the MFLAG value has a twofold function. First, as indicated above, it indicates that the cell is intersected by an interface. Second, it links the k index of the cell to an index in those arrays which store the mass, density, specific internal energy, and velocity components of each material in the interface cell.

Given m = MFLAG(k) - 100, the mass of material package n in cell k is stored in XMASS (n,m); similarly the density, specific internal energy, and velocity components of material package n are stored in RHO (n,m), SIE (n,m), US (n,m) and VS (n,m), respectively. In this way the k index of the cell

arrays are linked to the m index of the material arrays. Furthermore, the cell arrays are defined by summing over the materials in the interface cell as follows:

$$AMX(k) = \sum_{n} XMASS(n,m) \tag{4.4}$$

$$U(k) = \sum_{n} US(n,m) \cdot XMASS(n,m) \Big/ \sum_{n} XMASS(n,m) \tag{4.5}$$

$$V(k) = \sum_{n} VS(n,m) \cdot XMASS(n,m) \Big/ \sum_{n} XMASS(n,m) \tag{4.6}$$

$$AIX(k) = \sum_{n} SIE(n,m) \cdot XMASS(n,m) \Big/ \sum_{n} XMASS(n,m). \tag{4.7}$$

Listed in the table below are the arrays which store cell quantities and the arrays which store material quantities within the interface cells. Throughout this document these will be referred to as cell arrays and material arrays, respectively.

QUANTITY	CELL ARRAY	MATERIAL ARRAY
Mass	AMX	XMASS
Specific Internal Energy	AIX	SIE
Radial Velocity	U	US
Axial Velocity	V	VS
Density	(Not stored)	RHO
Pressure	P	(Not stored)
Shear Stress	STRSRZ	(Not stored)
Radial Stress Deviator	STRSRR	(Not stored)
Axial Stress Deviator	STRSZZ	(Not stored)

4.2.2 Definition of Material Arrays

Three subroutines, NEWFLG, NEWMIX, and NEWRHO are called to define the material arrays for a cell k that is being converted from a pure to an interface cell, which occurs when a material interface initially enters a pure cell.

First NEWFLG is called to find an unused location, m, in the material arrays. The convention which denotes that a material array location m is not being used is that RHO(1,m)=-1. Having found an unused location (HELP performs an error exit if no such location is found), NEWFLG redefines the value of MFLAG(k) to be m + 100.

In the case when an interface cell becomes a pure cell, which occurs when all material interfaces leave the cell, it is necessary to release the members of the material arrays which have stored the material properties for that interface cell. This process, which is implemented in TPHASE, consists of initializing the particular material variables to zero and setting RHO(1,m) = - 1, which denotes that the material variables whose second index is m are unused and can be assigned to another interface cell by NEWFLG. The process of converting an interface cell to a pure cell is completed by setting the cell's MFLAG value to the package number of the material that now fills the cell, or to zero if the cell has become completely void.

NEWMIX, after calling NEWFLG, defines the material arrays for the material which is currently in the cell. If mo was the value of MFLAG(k) on the previous cycle, when cell k was pure, and m is the location in the material arrays assigned to cell k by NEWFLG [m + 100 = MFLAG(k)], then NEWMIX defines the material variables for the material in the cell as follows (providing cell k was not a pure void cell, i.e., mo $\neq$ 0):

$$
\begin{aligned}
\text{XMASS}\ (mo, m) &= AMX(k) \\
\text{SIE}\ (mo, m) &= AIX(k) \\
\text{US}\ (mo, m) &= U(k) \\
\text{VS}\ (mo, m) &= V(k) \\
\text{RHO}\ (mo, m) &= AMX(k) \Big/ \text{CELL VOLUME} \quad .
\end{aligned}
$$

If mo = 0, the cell was previously a void cell and has become a free surface cell, since it contains a material interface. In that case RHO (NVOID, m) is set equal to 1, signifying that the cell is a free surface interface cell.

Any interface, of course, is a boundary between two materials, or one material and a void. Therefore, when a cell becomes an interface cell, both materials or the void have to be identified. The identity of one material, or void, is indicated by the flag of the cell when it was pure (mo in the example above). The other material or void is identified when the code senses which package tracer particles are intersecting the cell's boundaries. If that other package is the void, then RHO(NVOID,m) is set to 1. Otherwise subroutine NEWRHO is called to assign a density to the other material. NEWRHO picks the density from the neighbor cells nearest the intercept of the interface with the cell boundary. This density value is used, in most cases, as a flag indicating which other material interface has entered the cell. Since the densities of multimaterial interface cells are redefined in the pressure iteration (see Section 4.6.1) at the beginning of the next cycle, the value chosen need only be an approximation of the density of the material about to be transported into the cell. Only if the interface cell had been a pure void cell (mo = 0) and will contain only the new material which is entering the cell does the density defined by NEWRHO remain unmodified by the pressure iteration. This is the only case in which the density assigned to the material by NEWRHO is used as a real measure of the density of the material in the interface cell.

It is important to note that the material density array, RHO, is used as a flag as well as a material property. RHO(n,m)>0 indicates that the interface of material package n intersects the m interface cell. This flag must be set before any mass of package n can be transported into interface cell m. However, it is possible for the density variable, RHO (n,m), to be non-zero and the corresponding mass variable, XMASS (n,m), to be zero. In that case, the non-zero density only indicates that the interface of material package n is intersecting the cell.

On the other hand, when the material n interface leaves the m interface cell, the density variable, RHO (n,m), is set to zero to indicate that all the mass of material n should be "evacuated" from the cell on that cycle. In this case the density variable, RHO (n,m), being zero and the corresponding mass variable, XMASS(n,m), being nonzero signifies that any influxes of that material into the cell should be set to zero and the outfluxes adjusted so that their sum exactly equals the remaining mass. This adjustment of the transport terms is done by subroutine DMADJ and is discussed in more detail in Section 4.4.

4.2.3 Accounting for Subcycles

INFACE usually is subcycled to minimize the transport noise which occurs when an interface leaves a cell. The routine is executed two or more times each cycle using a fraction of the time step on each subcycle. It is therefore possible to create an interface cell after one or more subcycles of INFACE have been completed. In that case the mass transport variables (SAMMP, SAMMY, SGAMC, SAMPY) for that cell have not been accumulated for the completed subcycles and they need to be computed before adding in the transport

terms for the current and subsequent subcycles. Therefore, when INFACE has completed at least one subcycle and storage for the new interface cell has been set up, NEWMIX calls DMCALC, where the computation of transport terms is performed.

4.3 CALCULATION OF CELL FACE AREAS FOR TRANSPORT ACROSS INTERFACE CELL BOUNDARIES

The computation of fractional cell boundary areas is performed in order to use the material interface positions to define the transport of a given material across an interface cell boundary. For pure cells the mass transport between two cells is

$$\Delta m = \rho^d \bar{v} A \Delta t_{CDT} \tag{4.8}$$

where ρ^d is the donor cell density, $\bar{v}$ is an average of the cell-centered velocities of the two cells, A is the area of the cell boundary across which the mass is being transported and Δt_{CDT} is the time step* computed by CDT.

For interface cells this equation becomes

$$\Delta m_n = \rho_n^d \bar{v}_n A_n \Delta t_{INFACE} \tag{4.9}$$

where the density, ρ_n^d, and velocity, $\bar{v}_n$, are based on material quantities (RHO, US, VS) rather than cell quantities (AMX, U, V). The area term, A_n, is associated with material n and is determined by the position of the material n interface, while the time step, Δt_{INFACE}, is the time step for the cycle computed by CDT, divided by the number of passes through subroutine INFACE (see Eq.4.3). Thus the code computes a mass transport term for each material which intersects any boundary of an interface cell.

---

* See DT in Section 10.3.

This section describes the procedures used to compute the cell face area terms which are used to transport material across the boundaries of interface cells.

4.3.1 Defining Fractional Areas of Intersected Cell Faces

The cell face area terms associated with each material in an interface cell are computed in subroutine FRACS. The method given below for computing these areas puts no restrictions on how many interfaces are in a cell nor on how many times a single interface crosses a cell boundary. It does, however, assume that a material interface never intersects itself, and experience has shown that the logic of the code breaks down if it does. This is not a limitation of the method since the boundary will not cross itself unless the tracers become too sparse, and this can be prevented by periodically adding new tracers in subroutine ADDTCR. (See Section 8.2)

The method for computing the area terms will be better understood if the following assumptions and conventions are kept in mind. First of all, FRACS computes and stores only the area terms for the right and top boundaries of each interface cell, since the cells below and on the left provide the area terms for the other two boundaries. FRACS stores these area terms for interface cells in two arrays, FRACRT and FRACTP, for the right and top cell boundaries, respectively. These arrays are doubly dimensioned; given FRACRT(n,m) or FRACTP(n,m), the first dimension, n, gives the material package number, and the second dimension, m, is the index that links the interface cell to the material arrays by the relation m = MFLAG(k) - 100. Also, because the area term of a given material package appears in the mass transport equation, it must be defined for the right and top boundaries of every interface cell, even for those boundaries not intersected by a material interface.

Let us first consider how FRACS computes the area terms for the cell boundaries that are intersected by a material interface. The set of tracer particles which circumscribe a given material are considered consecutively (two at a time) by FRACS. (TX and TY are the FORTRAN variable names for the particles' x and y coordinates, respectively. The TX and TY arrays are doubly dimensioned; the first dimension gives the material package number, the second gives the sequential ordering of the particles. For simplicity the coordinates of the tracers are in grid line units rather than centimeters, since in grid line units the integer part of the coordinates indicates whether a pair of tracers straddles one or more cell boundaries.) If a pair of material n tracers straddles a cell boundary, FRACS computes the intercept of that boundary with the line between the two tracers. Given that point of intersection FRACS defines the area term for material n at that cell boundary to be the fractional cell boundary area on one side of the intercept. To determine which side of the intercept to associate with material n, FRACS uses the convention that material n is on the left as one proceeds from the first to the second tracer of the pair.

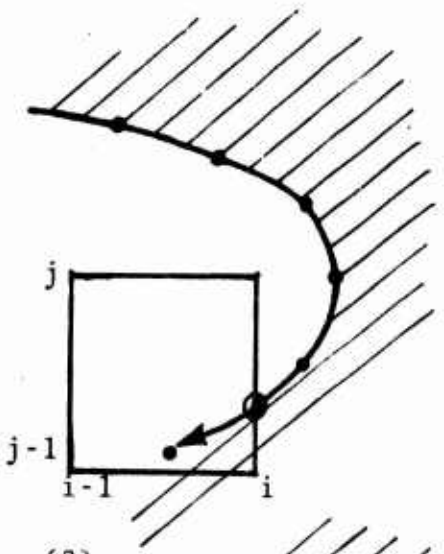
For the cell boundary not intersected by a material interface the area term for a given material is either the total cell boundary area or is zero. The following discussion illustrates how the area terms of these non-intersected cell boundaries are correctly defined by "presetting" them when an interface enters a cell and "resetting" them when it leaves.

4.3.2 Presetting Non-Intersected Cell Face Areas

The sequential order of the tracer particles determines whether an interface is entering a cell or leaving it. When the

interface is entering a cell, and is crossing this cell boundary for the first time, the program processes the other boundaries of the cell in a clockwise order, presetting the corresponding area terms to the total area of that cell boundary, until it encounters a side that has already been crossed by the same material interface (i.e., a side which has an associated area term that is non-zero and is less than the total area). The first three cases diagrammed below illustrate the presetting procedure.

Case (1)



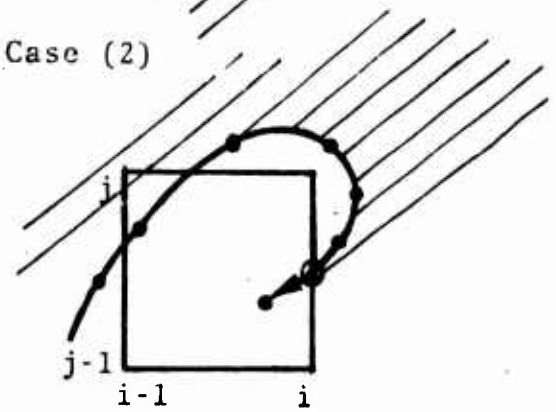
Area terms preset:

$$\text{Bottom} = \pi \left(x_i^2 - x_{i-1}^2 \right)$$

$$\text{Left} = \left(y_j - y_{j-1} \right) 2\pi \, x_{i-1}$$

$$\text{Top} = \pi \left(x_i^2 - x_{i-1}^2 \right)$$

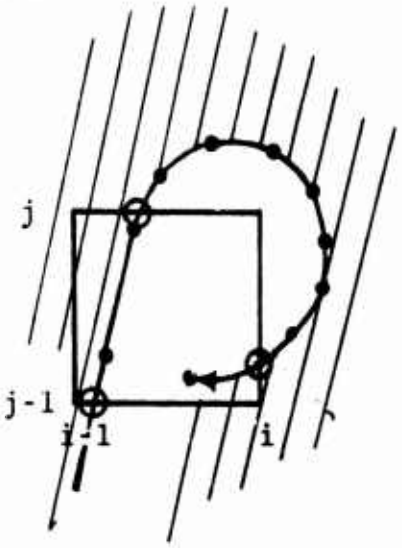
Case (2)



Area terms preset:

$$\text{Bottom} = \pi \left(x_i^2 - x_{i-1}^2 \right)$$

Case (3)

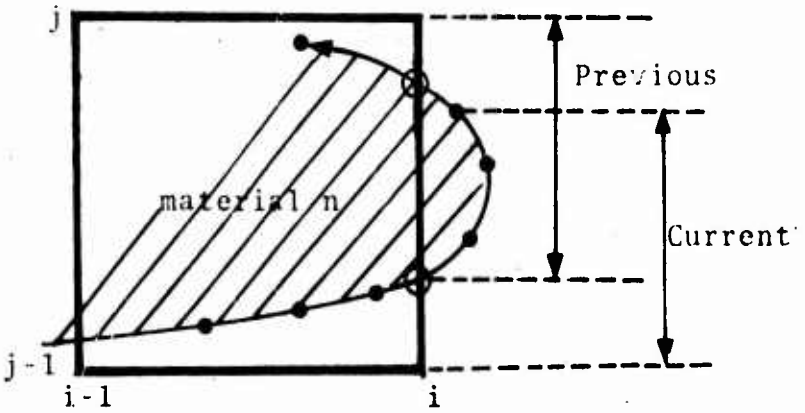


Area terms preset:

None - the bottom boundary has been intersected previously by the same interface.

In order that there be no restriction on the number of times the material interface crosses a given cell boundary, the presetting procedure is generalized in the following manner. If the interface of material n crosses a cell boundary for the second time, the code does not preset the area terms of the other sides of the cell if material n lies between the two points of intersection. This is illustrated by Case 4.

Case 4



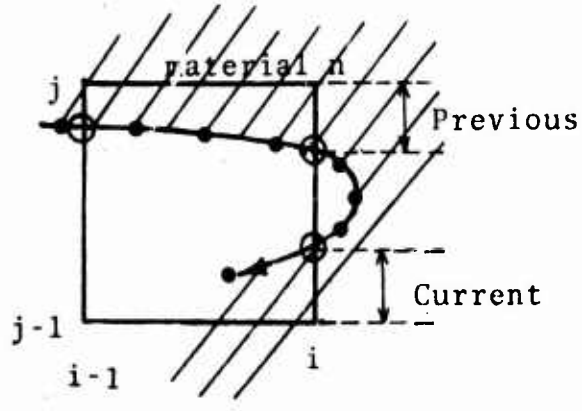
Area terms preset:

None - material n lies between the two intersection points.

The pro[illegible]m senses that material n is between the points when [illegible]n of the area term computed on the previous cross[illegible](s) and the one computed currently is greater than the total area of the cell face.

If, as in Case 5, the sum of the area term computed on the previous crossing(s) and the one computed currently is <u>less</u> than the total area, material n is outside the two intercepts and the procedure described above for presetting the area terms of the other sides of the cell is followed.

Case (5)

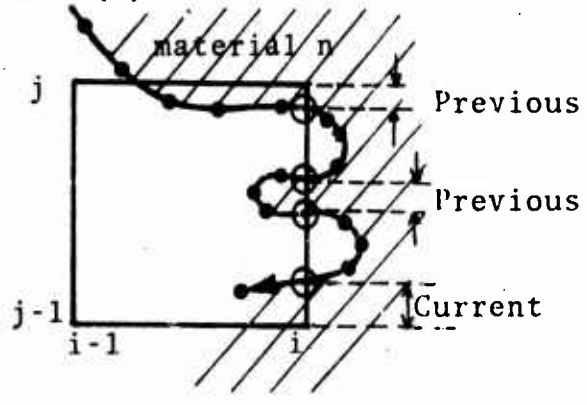


Area terms preset:

$$\text{Bottom} = \pi \left(x_i^2 - x_{i-1}^2 \right)$$

It should be noted that these conventions apply regardless of how many times the interface crosses the boundary, as illustrated by Case (6) and Case (7).

Case (6)

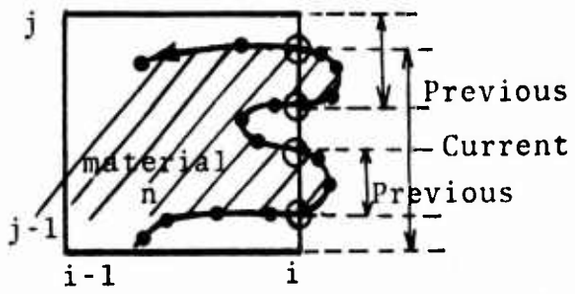


Area terms preset:

$$\text{Bottom} = \pi \left(x_1^2 - x_{i-1}^2 \right)$$

$$\text{Left} = \left(y_j - y_{j-1} \right) 2\pi x_{i-1}$$

Case (7)



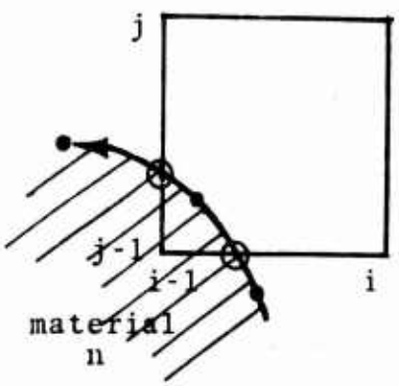
Area terms preset:

None

4.3.3 Resetting Non-Intersected Cell Face Areas

The procedure for resetting the area terms when the interface leaves a cell is the following. If the interface is crossing the cell boundary for the first time, the program processes the other faces of the cell in a clockwise order and resets their area terms to zero, until it encounters a side that has been crossed previously by that same material interface. This procedure is illustrated by Cases (8), (9), and (10) below.

Case (8)

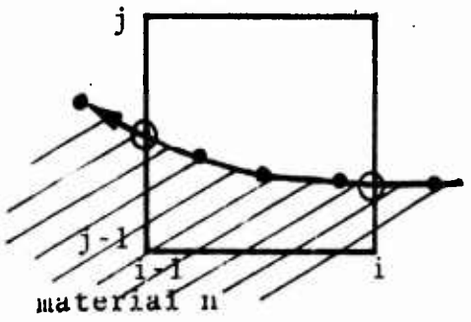


Area terms reset:

Top = 0

Right = 0

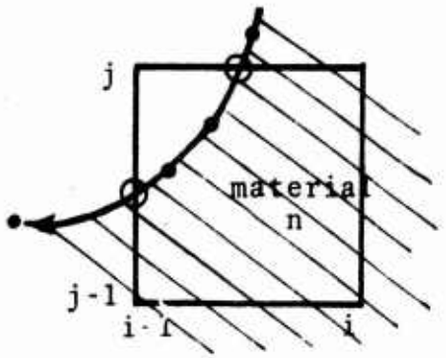
Case (9)



Area terms reset:

Top = 0

Case (10)

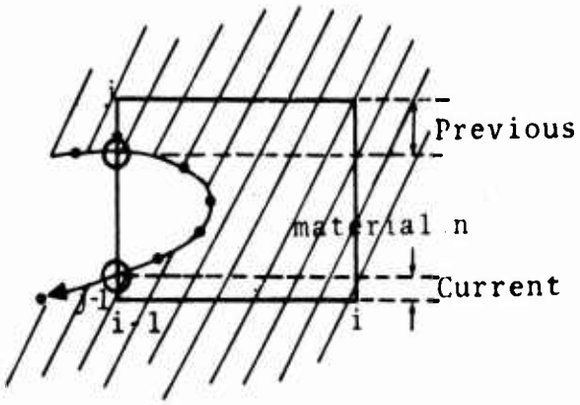


Area terms reset:

None

In Cases (11) and (12) below, the side where the interface leaves has been previously crossed by this interface. The program sums the area terms resulting from any previous crossings of that boundary and the current crossing. If, as in Case (11), the sum of these areas is less than the total cell boundary area, the material is outside the two intercepts and none of the area terms of the other sides are set to zero.

Case (11)

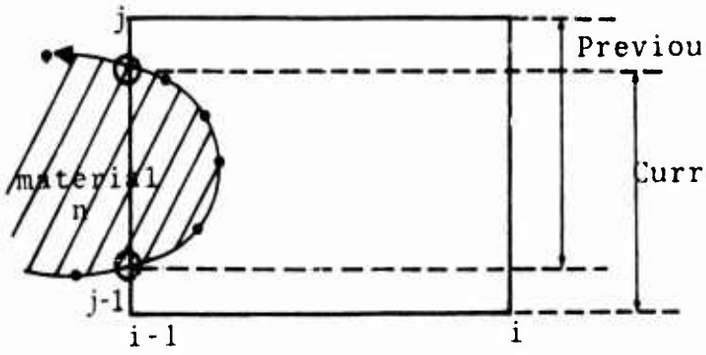


Area terms reset:

None

If, however, the sum is greater, as in Case (12), the material is between the intercepts and the program proceeds to reset to zero the areas of the other non-intersected sides of the cell.

Case (12)



Area terms reset:

Top = 0

Right = 0

Bottom = 0

4.4 ADJUSTMENT OF INTERFACE CELL MASS TRANSPORT TERMS

The four mass transport terms for interface cells (one for each cell face) are computed by DMCALC using material densities and velocities and the cell face area terms associated with the material packages as computed by subroutine FRACS. However, subsequent to this calculation, DMADJ is called to (possibly) modify these transport terms. There are two reasons the

transport terms may need to be adjusted: one, to exactly empty a cell of a material when its interface leaves the cell, i.e., evacuation; and two, to prevent the outflux of material from exceeding the mass of the material in the cell, i.e., over-emptying.

4.4.1 Evacuation of a Material

To signify that the material n interface has passed out of cell k the density of material n, RHO(n,m), is set to zero. This zero density is used as a signal to subroutine DMADJ to adjust the transport terms (one for each cell face) of material n so that cell k will be exactly evacuated of material n. This adjustment of the four transport terms usually occurs twice in the computational cycle; once at the beginning of the first INFACE subcycle (in case the material r interface left the cell on the previous cycle*) and once after all INFACE subcycles are completed (in case the material n interface moved out of cell k during or after the first, and before the last, INFACE subcycle). For the first case the four transport terms from the previous cycle are used to indicate the direction of the flow and the direction of the evacuation. In the second case, the four transport terms just computed by DMCALC are used to determine how material n is to be evacuated.

If material n is to be evacuated from cell k, then all influx transport terms of material n are set to zero, and each outflux term is adjusted in proportion to its fraction of the total outflux computed by DMCALC. If the total outflux computed by DMCALC is zero, the material is removed from the grid (evaporated) at the end of TPHASE.

---

*The tracer particles are moved at the end of each INFACE subcycle. If the interface moves out of a cell on the last subcycle of INFACE, the code does not sense this until the first pass through INFACE on the next computational cycle.

These evacuation procedures are used for a cell that has lost one interface, but is still cut by another interface, as well as for cells that have become pure. After the area terms associated with each material package have been computed, FLGSET searches for cells that were interface cells on the previous cycle but are no longer cut by an interface. Such a cell has become pure and its flag, MFLAG(k), is made negative to signify that fact until the transport has been completed in TPHASE. To determine what material package a newly pure cell k belongs to, FLGSET first looks for a neighbor that is pure and assumes cell k will belong to the same package as one of its pure neighbors. If, however, all of its neighbors are interface cells, the top cell face area terms of the cell below* kb, are examined. If the cell face area for material n at the top of cell kb is equal to the total cell face area, then cell k must be inside the material package n boundary. Therefore, all material densities for cell k except that for material n are set to zero, and at the end of TPHASE cell k becomes a pure material package n cell with MFLAG(k) = n.

4.4.2 Overemptying of a Material

After the subcycles of INFACE are completed, subroutine DMADJ is called to check that the outflux of each material in each interface cell does not exceed its current mass. If the sum of the outgoing transport terms of material n is greater than the current mass of material n, the outgoing transport terms of material n are reduced proportionately, as in the evacuation procedure described above.

---

*The cell on the left of cell k is used analogously if cell k is in the bottom row of the grid.

4.5 TYPES OF INTERFACE CELLS

There are several types of interface cells. In the following paragraphs they will be defined and then discussed in terms of special prescriptions that apply to them.

4.5.1 Multimaterial Cells

Most interface cells contain more than one material, and the only practical restriction on the number of materials in any one cell is the array dimensions imposed by the user* A detailed description of the determination of material properties in multimaterial cells is given in Section 4.6. The following is a summary.

The partial volumes of the materials in a multimaterial cell are determined when material pressures are equilibrated. The densities of the materials are adjusted in an iteration until the pressures of the materials equilibrate. (See Section 4.6.1)

The shear yield strength of a multimaterial cell is a volume weighted average of the shear yield strengths of its constituents. The code assumes, however, that a multimaterial cell has no tensile strength, i.e., negative pressures are set to zero in multimaterial cells. (See Section 4.6.2)

The internal energy increment of a multimaterial cell is partitioned by partial masses in SPHASE and HPHASE. The internal energy is updated in TPHASE to account separately for the transport of internal energy and the thermalization of kinetic energy of each material in the cell. (See Section 4.6.3)

4.5.2 Free Surface Cells

Any cell that is intersected by the void interface is regarded as a free surface cell, and is flagged via the RHO array: RHO (NVOID,m) = 1, where NVOID is the number of material

---

* The MFLAG convention, however, imposes a limit of 99 material packages in a given problem.

packages plus one (NMAT+1), and m = MFLAG(k)-100. A free surface cell can contain more than one material as well as the void, although frequently free surface cells do contain only one material.

In CDT, if the equilibrated pressure of a multimaterial, free surface cell is less than PMIN, a user-controlled input number (See Section 7.2.1), the densities of the materials in the cell are not modified unless the sum of their partial volumes exceeds the volume of the cell. This rule prevents material densities in a free surface cell from being based on the mass being extended over the entire cell volume. However, if the free surface cell has only one material, the pressure iteration is bypassed and the density, used in the equation of state, RHOW, is the total cell mass divided by the total cell volume. If the cell pressure based on this density is negative, it is set to zero and the density of the material, RHO, is redefined only if it is less than the assumed density, RHOW. (See also Section 4.2.2.)

In SPHASE, the free surface cells are assumed to have no strength. Therefore the deviator and shear stresses of a free surface cell are zero.

In DMCALC, when computing the mass transported from a free surface cell into a non-free surface cell, the acceptor cell density instead of the donor cell density is used, since, in general, the densities of the non-free surface cells are better determined.

4.5.3 Slipline Cells

Sliplines in HELP are not restricted to grid lines or cell diagonals. They do coincide with material interfaces, however. Therefore a cell that contains a slipline is also an interface cell.

The materials in a slipline cell are accelerated in inverse proportion to their densities in HPHASE. (See Section V.) In SPHASE the assumption is made that a slipline cell has no strength, therefore its deviator and shear stresses are set to zero.

The velocity components of the "slave" and "master" materials in a slipline cell are different. The cell velocity components represent a mass weighted average of the material velocity components. These cell velocities are used in computing the work terms in HPHASE and SPHASE and in moving the material and passive tracers in MOVTCR. The material velocities are used to compute the mass and momentum transport terms in TPHASE.

4.6 DETERMINATION OF MATERIAL PROPERTIES IN MULTIMATERIAL CELLS

4.6.1 Pressure and Density Iteration for Multimaterial Cells

For one-material cells, it is possible to compute the pressure directly from the material density and specific internal energy. However, for cells containing more than one material, it is necessary to make additional assumptions or observations in order to determine the cell pressure. In the present method we determine the cell pressure for multimaterial cells by an iteration procedure. Specifically, the densities of the various materials within the cell are varied, subject to the condition that the cell be exactly filled by the masses therein, until the individual pressures converge to a common value, which is taken to be the cell pressure. Specific internal energies are held constant during the iteration. This process gives, in addition to the cell pressure, the densities of the individual materials for subsequent use in the DMCALC and TPHASE transport calculations.

The procedure for determining the pressure and the densities in a multimaterial cell is actually divided into

two parts: a pre-iteration calculation to insure that the cell is filled exactly; and an iteration calculation to converge on the desired P's and ρ's.

4.6.1.1 Pre-iteration Calculation

In general, the masses and densities which exist at the beginning of the calculation do not exactly fill the cell, since cell masses are changed in the TPHASE calculation of the previous cycle. The pre-iteration calculation fills the cell exactly, taking into account the (computed) compressibilities of the various materials in the necessary alteration of densities. Stepwise, the code

1. Computes P_i from the old value of ρ_i for each material within the cell (E_i is assumed constant throughout).
2. Varies the ρ_i by one percent to compute new P_i.
3. Determines the constant-energy compressibilities, called C_i^2, for each material

$$C_i^2 = \Delta P_i / \Delta \rho_i \tag{4.10}$$

   where the ΔP_i and $\Delta \rho_i$ are given from the results of (1) and (2).
4. Normalizes (see proof below) to fill cell exactly:

$$\Delta V_i = -\Delta P / h_i^2 \tag{4.11}$$

   where

$$h_i^2 = \left(\rho_i \, C_i \right)^2 \qquad (4.12)$$

and

$$\Delta P = \frac{(VOL-VCELL)}{\sum \frac{m_i}{h_i^2}} \qquad (4.13)$$

$$VOL = \sum \frac{m_i}{\rho_i} \qquad (4.14)$$

with m_i = mass of constituent i, and
VCELL = volume of the cell.

With the above increments in specific volume, the new specific volumes are given by

$$V_i = V_{i,old} + \Delta V_i \quad . \qquad (4.15)$$

Then,

$$\rho_i = 1/V_i \qquad (4.16)$$

gives the desired new densities which cause the cell to be filled exactly.

It is easy to see that the calculation defined above causes the cell to be filled exactly. To show this,

$$m_i V_i = m_i \left(V_{i,old} + \Delta V_i \right)$$

$$= m_i / \rho_{i,old} - m_i \, \Delta P / h_i^2$$

$$= m_i / \rho_{i,old} - \frac{m_i (VOL - VCELL)}{h_i^2 \sum \frac{m_j}{h_j^2}} \quad . \qquad (4.17)$$

Summing,

$$\sum_i m_i V_i = VOL - (VOL - VCELL) \frac{\sum \frac{m_i}{h_i^2}}{\sum \frac{m_j}{h_j^2}}$$

$$= VCELL \qquad (4.18)$$

showing that the new volumes $m_i V_i$ of the constituents add up to the total cell volume, as desired.

4.6.1.2 Iteration Calculation

Given $P_i = f_i \left(V_i, E_i \right)$ for materials i= 1,...,M, the cell pressure $\bar{P} = P_1 = P_2 = \ldots = P_M$ is found by varying the V_i until the P_i are equal within some specified accuracy, subject to the conditions that the cell remain exactly filled and that the E_i are constants.

The equation for volume conservation is

$$\sum m_i \, \Delta V_i = 0 \qquad (4.19)$$

and the equation that causes material i to undergo a change in specific volume ΔV_i, such that its pressure changes from

its current value $P_i^n = f_i\left(V_i, E_i\right)$ to a value $\overline{P}^{n+1}$, common to all materials in the cell at the end of the iteration cycle, is

$$\Delta V_i = \left(\overline{P}^{n+1} - P_i^n\right)\left(\frac{\partial V_i}{\partial P_i}\right)_{E_i} \quad i = 1,\ldots,M \qquad (4.20)$$

These equations, for $i = 1,\ldots,M$ can be regarded as M+1 equations for the M+1 quantities ΔV_i and $\overline{P}^{n+1}$. The other quantities in the equations are either known constants (m_i) or are updated each cycle of the iteration [the P_i^n and the $\left(\partial V_i/\partial P_i\right)_{E_i}$] and are taken to be constants while solving these equations for ΔV_i and $\overline{P}^{n+1}$. The solutions are

$$\overline{P}^{n+1} = \frac{\sum P_i\, w_i^n}{\sum w_i^n} \qquad (4.21)$$

$$\Delta V_i = \frac{1}{h_i}\left(P_i^n - \overline{P}^{n+1}\right) \qquad (4.22)$$

where

$$h_i = \left(\partial P_i/\partial V_i\right)_{E_i} \qquad (4.23)$$

$$w_i = m_i/h_i \quad . \qquad (4.24)$$

That these equations are solutions to the given equations can be verified by direct substitution.

The iteration would be exact (single cycle convergence) if the input coefficients $\left(\partial P_i/\partial V_i\right)_{E_i}$ were constants, since

the solution does not involve additonal approximations. In the iteration, these coefficients are refined each step of the iteration by making use of the P_i, V_i points which were generated during the previous step. Further, some pains are taken to determine realistic starting values of these coefficients, in the pre-iteration calculation, in the belief that this procedure will hasten convergence sufficiently to reduce total computing time.

Stepwise, the iteration calculation is as follows:

1. Compute new $\overline{P}^{n+1}$ and ΔV_i by the above formulas, using known values of P_i^n and $\left(\partial V_i/\partial P_i\right)_{E_i}$.
2. Compute new V_i^{n+1} by adding the above ΔV_i to the old specific volumes.
3. Compute new P_i^{n+1}, using the updated V_i.
4. Compute refined values of the coefficients $\left(\partial V_i/\partial P_i\right)_{E_i}$ by using this last point [from (2) and (3) above] and that previous P_i,V_i point (two such points are saved for each material during the iteration) which is closest in pressure to this last point, i.e.,

$$\left(\frac{\partial V_i}{\partial P_i}\right)_{E_i}^{n+1} = \frac{V_i^{n+1} - V_i'}{P_i^{n+1} - P_i'} \qquad (4.25)$$

   where P_i', V_i' is the closest previous point.
5. Return to (1), replacing the old values of the P_i^n and the $\left(\partial V_i/\partial P_i\right)_{E_i}$ with the updated values.

The iteration is completed when the P_i all equal $\bar{P}$ to some preset accuracy, PRCNT (See Section 7.2.1). In most experience to date, convergence has been obtained in two or three steps, where a convergence criterion $|P_i - \bar{P}| < 10^{-3} \bar{P}$ was used.

4.6.1.3 Equation of State Modifications for the Iteration Procedure

The Tillotson equation of state (described in Section 2.3.1.2) has been used in most work to date with HELP and is sufficiently general to provide a satisfactory description of most media which is initially condensed and which may vaporize as a result of heating. However, for some materials which are sufficiently expanded and cold ($E < E_s$), the slope of a constant energy curve (in the P-V plane) is positive. For pure cells, these meaningless states are avoided by replacing the quantity ($A\mu + B\mu^2$) with its minimum value (a negative pressure) if the cell density is less than the density corresponding to this minimum value.

To assure convergence of the pressure iteration in multimaterial cells, it is necessary to have a continuous equation of state and for $\partial P/\partial V$ to be everywhere negative. To this end the term

$$A\mu + B\mu^2 \tag{4.26}$$

is replaced by

$$A\mu + h(V)B\mu^2 \tag{4.27}$$

where

1. $h(V) = +1$ for $V \leq V_o$ (compressed states)
2. $h(V)$ decreases linearly to -1 in the interval

$$V_o < V < V_o \frac{2B}{2B-A} \tag{4.28}$$

and

3. $h(V) = -1$ for $V \geq V_o \frac{2B}{2B-A}$.

The effect of these modifications is to change B to -B in a continuous manner and hence insure that $\partial P/\partial V$ is always negative. Negative pressures that may result after the iteration is completed are set to zero, since tensile states in interface cells are assumed not to exist. In view of this latter assumption, the indicated equation of state changes do not normally affect the physical result, since they alter only states that would ordinarily be tensile. The purpose of the changes is to make the equation of state well-behaved, so that the iteration converges properly.

4.6.2 Shear Yield Strength of Multimaterial Cells

The shear yield strength of a multimaterial cell is a volume weighted average, $\bar{Y}$, of the shear yield strengths of its constituents, Y_n:

$$\bar{Y} = \sum Y_n V_n / \sum V_n \quad . \tag{4.29}$$

If, however, one of the constituents is an ideal gas or a high explosive, (MAT(n) $\geq$ 20), the shear yield strength of

the cell is set to zero. Likewise, if the cell contains the void interface (RHO(NVOID,m) = 1) or a slipline interface (THETA(m) $\geq$ 0), with m = MFLAG(k) - 100, the shear yield strength of the cell is set to zero. And finally, the shear yield strength of the cell is zero if the density of any one of its constituents is below the density corresponding to the maximum tensile strength of the constituent (RHO(n,m) $< \rho_{o_n}$ · AMDM(n)). This presumes that a material that has failed in tension has no shear yield strength.

4.6.3 Partitioning of Internal Energy Increments in Multimaterial Cells

The internal energy increments, ΔE_I, computed by HPHASE and by SPHASE are partitioned amongst the materials in a multimaterial cell in proportion to their masses. For example, the internal energy increment for material n is

$$\Delta E_{I_n} = \Delta E_I \cdot m_n / \sum_{j=1}^{NMAT} m_j \quad . \qquad (4.30)$$

In other words the HPHASE and SPHASE gain in specific internal energy is the same for all the materials in the cell.

The situation is somewhat more complex in TPHASE, since the internal energy increment is due not only to transport but also to the thermalization of kinetic energy. It is natural to imagine that the interacting materials (all of the masses of material n within the cell at the end of the time step) have collided in a coordinate system in which their center of mass has zero velocity.

First, we recall that the new velocity components for material n are chosen to conserve momentum

$$u_n^{(n+1)} = \left[m_n^{(n)} u_n^{(n)} + \sum_i \delta\left(m_{in} u_{in}^{(n)} \right) \right] / \left(m_n^{(n)} + \sum_i \delta m_{in} \right) \qquad (4.31)$$

$$v_n^{(n+1)} = \left[m_n^{(n)} v_n^{(n)} + \sum_i \delta\left(m_{in} v_{in}^{(n)} \right) \right] / \left(m_n^{(n)} + \sum_i \delta m_{in} \right) \qquad (4.32)$$

and that the new mass of material n is

$$m_n^{(n+1)} = m_n^{(n)} + \sum_i \delta m_{in} \quad . \qquad (4.33)$$

Here the subscript i denotes the four sides of the cell; the subscript n denotes the material package, and the superscript (n) denotes pre-TPHASE quantities. The velocity associated with each of the transport masses is the donor cell material velocity; e.g., if a mass Δm_{in} entered the cell under consideration through side 1, the associated velocity u_{in} is the velocity of material n in the cell from whence the mass came.

We note first that $u_n^{(n+1)}$, $v_n^{(n+1)}$ as determined to conserve momentum by Eqs. (4.31) and (4.32) above, are the velocities of the center of mass of the material n in the cell at the end of the time step. This means that a component mass δm_{in} with pre-TPHASE velocities, $u_{in}^{(n)}$, $v_{in}^{(n)}$, has the following amount of kinetic energy thermalized

$$\frac{1}{2} \delta m_i \left[\left(u_n^{(n+1)} - u_{in}^{(n)} \right)^2 + \left(v_n^{(n+1)} - v_{in}^{(n)} \right)^2 \right]$$

in the center of mass coordinate system. Thus the total thermalized KE for material n is

$$TKE_n = \frac{1}{2} m_n^{(n)} \left[\left(u_n^{(n)} - u_n^{(n+1)} \right)^2 + \left(v_n^{(n)} - v_n^{(n+1)} \right)^2 \right]$$

$$+ \frac{1}{2} \sum_i \delta m_{in} \left[\left(u_{in}^{(n)} - u_n^{(n+1)} \right)^2 + \left(v_{in}^{(n)} - v_n^{(n+1)} \right)^2 \right] . \quad (4.34)$$

Some interpretation of this equation clarifies its meaning: the first term, plus those terms from the summation for which $\delta m_{in} < 0$ (material leaving the cell with donor cell velocities $u_{in}^{(n)} = u_n^{(n)}$, $v_{in}^{(n)} = v_n^{(n)}$) represents the thermalized KE for that fraction of the material which remains in the cell. The additional terms ($\delta m_{in} > 0$ in the summation) represent the thermalized KE associated with that material which came into the cell during the time step. For these terms the $u_{in}^{(n)}$, $v_{in}^{(n)}$ refer to the velocities of material in the neighbor cells which acted as donors.

The change in internal energies is again given by

$$\Delta E_{I_n} = \sum_i \Delta E_{I_{in}} + TKE_n \quad . \quad (4.35)$$

In other words, material n gains an amount of total internal energy which is that due to transport, plus the amount of thermalized kinetic energy for material n.

The above discussion assumes that the material velocities are not equal to the cell velocities, i.e., that the cell contains the slipline. If the cell does not contain a slipline, equations (4.33) through (4.35) still hold, only the material velocities are defined using cell quantities as follows:

$$u_n^{(n+1)} = \left[\left(m^{(n)}u^{(n)} + \sum_i \delta\, m_i u_i^{(n)}\right)\right] / \left(m^{(n)} + \sum_i \delta m_i\right) \tag{4.36}$$

$$v_n^{(n+1)} = \left[\left(m^{(n)}v^{(n)} + \sum_i \delta\, m_i v_i^{(n)}\right)\right] / \left(m^{(n)} + \sum_i \delta m_i\right) \tag{4.37}$$

i.e., the velocity components of all the materials in the cell will be the same and will be equal to the cell velocity components.

CHAPTER V

SLIPLINES

5.1 GENERAL COMMENTS

Sliplines have existed in Lagrangian codes for several years and some of our thinking and terminology comes from these earlier accomplishments. For example, in HELP one package is considered the "master" along which another "slave" package slips. However, an important difference between this current effort using an Eulerian code and the earlier Lagrangian models is that the shape of the slip surface is determined by the stress field and need not coincide with grid lines or node points.

The slipline in HELP is also a material package interface; therefore, the slipline cells are also interface cells. The angle of the slipline in a given slipline cell is measured from the x-axis of the grid, using the intercepts of the slipline interface with the cell boundaries. These intercepts are computed while processing the interfaces in FRACS and are stored by PTSAV. The stored intercepts are then used by THETAS to compute the angle of the slipline in each slipline cell.

Since the slave and master materials in a slip cell develop different velocity components tangent to the slipline, it is necessary to store distinct velocity components for each material. (In an earlier version of HELP, it was assumed that all materials in an interface cell would have the same velocity components.) Two material arrays, US, VS, have been added to store the material velocities in interface cells. To describe the angle of the slipline across a given slip cell, the THETA array was added. When THETA(m)=-1 (m = MFLAG(k)-100), the interface cell does not contain the slipline; when

THETA(m) $\geq$ 0 the slipline does cross the interface cell at an angle (in radians) equal to THETA(m).

It should be noted that more than one material package can be designated as a slave or master. Therefore it is possible for a slipline cell to contain more than one slave or master package. In that case, the velocities of all the slave materials in the cell are equilibrated as are the velocities of all the master materials. The slave density, ρ_s, used in the equations that follow is an average of the densities of all the slave materials in a given slip cell, likewise for the master density, ρ_m.

5.2 RELATIVE ACCELERATION OF MASTER AND SLAVE MATERIALS (UVCALC)

A cell that contains the slipline has, in a sense, failed. Therefore the code assumes the material in a slip cell has no shear yield strength and the deviator stresses of a slip cell are set to zero. The only stresses acting on a slip cell are simply the hydrostatic pressures. The acceleration of slip cells due to stresses is computed entirely in HPHASE. After updating the momenta of a slipline cell, HPHASE calls UVCALC to establish the relative acceleration of the master and slave materials. Given the post-HPHASE cell centered momenta (m(k)u(k), m(k)v(k)) and the angle θ of the slipline through cell k, the master and slave material velocities, us_m, vs_m, and us_s, vs_s, are updated by solving the following four simultaneous linear equations:

$$m_s\, vs_s + m_m vs_m = (m_s + m_m)\, v(k) \qquad (5.1)$$

$$m_s\, us_s + m_m us_m = (m_s + m_m)\, u(k) \qquad (5.2)$$

$$us_s \sin\theta - vs_s \cos\theta = us_m \sin\theta - vs_m \cos\theta \qquad (5.3)$$

$$\rho_s [(us_s - us_s') \cos\theta + (vs_s - vs_s') \sin\theta] \qquad (5.4)$$

$$= \rho_m [(us_m - us_m') \cos\theta + (vs_m - vs_m') \sin\theta]$$

where us_s', vs_s', us_m', vs_m' are pre-HPHASE slave and master material velocities. The updated material velocities must conserve the cell's axial and radial momenta (Eqs. (5.1) and (5.2)). The master and slave velocity components normal to the slipline must be equal (Eq. (5.3)), and the HPHASE change to the slave and master velocity components tangent to the slipline must be inversely proportional to the slave and master material densities, ρ_s, ρ_m (Eq. (5.4)).

Solving these four equations results in the thermalization of some of the slip cell's kinetic energy. This thermalized kinetic energy is converted to internal energy and is partitioned amongst the materials in the cell in proportion to their masses.

5.3 TRANSPORT OF MASTER AND SLAVE MATERIALS

The mass, momentum and energy transport terms are computed in DMCALC and TPHASE. Since some interface cells, namely those along the slipline, have different velocity components for the master and slave materials, the mass and momentum transport equations for interface cells must employ material velocities rather than cell velocities.

For example, the mass of material n to be transported across the i cell face is

$$\delta m_n = - \rho_n^d \, \overline{\omega}_{n_i} \, A_{n_i} \, \Delta t \qquad (5.5)$$

where ρ_n^d is the density of material n in the donor cell, $\bar{\omega}_{n_i}$ is the mass transport velocity based on the material n velocities of the donor and acceptor cells, and A_{n_i} is the i cell face area associated with material n. Likewise, the momentum transport terms are based on material velocities rather than cell-centered velocities:

$$\delta mu_n = \delta m_n \cdot us_n^d \quad . \tag{5.6}$$

$$\delta mv_n = \delta m_n \cdot vs_n^d \quad . \tag{5.7}$$

Since the material velocities equal the cell velocities in cells that do not contain a slipline, the equations above result in the same mass and momentum transport for non-slip interface cells as do equations that use cell-centered quantities.

5.4 MODIFICATION OF POST-TPHASE MASTER AND SLAVE VELOCITIES (UVMOD)

After the material velocities of a slip cell have been updated in TPHASE, they must be altered to insure that the material velocity components normal to the slipline are equal.

Subroutine UVMOD is called from TPHASE after all cells have been updated for the effects of transport. UVMOD searches for interface cells that contain a slipline (THETA(m) $\geq$ 0) and modifies the post-TPHASE master and slave material velocities of those cells so that their components normal to the slipline are equal, i.e., so that Eq. (5.3) in the discussion above is satisfied. The material velocities of non-slip interface cells are not modified by UVMOD.

5.5 USE OF SLIPLINES

Sliplines in HELP have been employed in a variety of situations; for example, at the high explosive-metal interfaces of shaped charge and fragmenting munitions calculations, and at the interface between the plug and target when modeling thin plate perforation.

When there is a large density difference between the master and slave materials (as between an expanded HE and a metal casing), the relative motion of the two packages is generated primarily by the HPHASE accelerations, which are density weighted and which satisfy Eqs. (5.1) through (5.4) in the discussion above. However, when the densities of the master and slave materials are close (as between a plug and a target), the relative motion of the packages results primarily from the different momentum transported to the master and slave materials in TPHASE. For example, given that the projectile and plug are both slave packages, and the target is a master package, the material velocities of the plug and projectile in a slip cell are equilibrated, thereby transferring the projectile momentum to the plug but not to the target.

CHAPTER VI

PLUGGING FAILURE

A plugging model has been incorporated into the HELP code which the user activates if he or she anticipates plugging failure. As the following discussion will clarify, the decision to use the plugging failure model must be made when the calculation is generated. As the model now exists, its application is restricted to the impact of right circular cylinders or truncated cones into homogeneous targets of finite thickness. Furthermore, the user must supply an empirically derived material parameter (PLWMIN) which determines the shape and mass of the plug that is formed

In the sections that follow, the procedures for generating a plugging calculation as well as the code's method for evolving the plug will be discussed.

6.1 GENERATION OF A PLUGGING CALCULATION

6.1.1 Designation of Material Packages

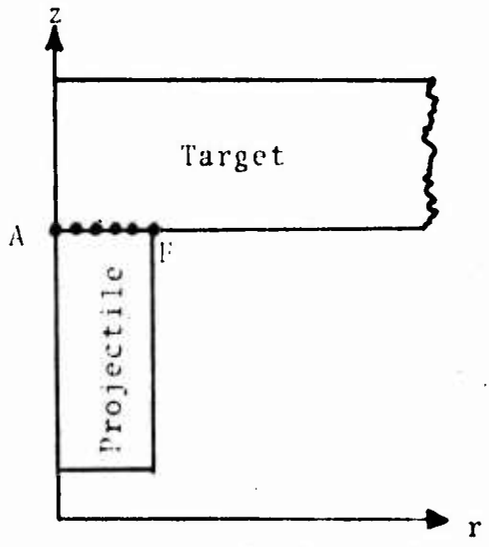
The plugging model assumes the projectile is package 1, the plug is package 2, and the target is package 3. The projectile can be made up of several material packages as long as the part of the projectile which will interact with the target is homogeneous and is designated as package 1. Furthermore, the projectile and plug packages must be "slave" packages and the target a "master" package.

6.1.2 Placement of Plug Tracer Particles

Unlike the other material packages in a HELP calculation, the dimensions of the plug package are unknown at the beginning

of the calculation. However, the user does specify in the input deck a set of plug tracer particles which later in the calculation are automatically redefined so as to circumscribe the plug package as it evolves. Initially these plug tracers coincide with the projectile and target tracers along the projectile-target interface. These initial tracer positions are based on the assumption that the vertical edge of the plug package begins at the top right edge of the projectile.

In the example below, the relationship between the projectile, target and plug tracer particles at time T = 0 is illustrated:



Projectile - The last six projectile tracer particles (excluding the dummy end point*) are at positions F through A.

* The last tracer of a package is a dummy point. It is used only to indicate the package has been completely circumscribed.

Target - The first six target tracer particles are at positions A through F.

Plug - The first six plug tracer particles are at positions A through F. The final six plug tracer particles (excluding the dummy point) are at positions F through A.

There is a set of n duplicate plug and target tracer particles placed at position F in order to define the vertical edge of the plug package as it is extended. (See Section 6.2.2.1.) These n duplicate particles are generated automatically by PLGADD, given the user-supplied definition of the plug slipline end-points (NBGSD(2), NENDSD(2)). The number of duplicate particles generated is (NENDSD(2) - NBGSD(2) + 1) and is therefore controlled by the user.

6.1.3 Placement of Passive Tracer Particles

The user must specify a region of the target in which the plug is expected to form. In each cell of that region, four passive tracer particles are automatically placed by SETUPA. These passive particles are used to track material within the target, to accumulate its plastic work, and to compute its angle of maximum shearing stress. This information is needed for the plugging failure criterion discussed in Section 6.2.1.

6.1.4 Slipline Specification

To simulate the action of a plug being pushed out of a target plate, it is necessary to generate slip surfaces between the plug and the target and between the deforming projectile

head and target lip. These slip surfaces do not exist, of course, until the plug evolves. In a plugging calculation the tracer particles that define the endpoints of the slip-line for each package are all initially at the top right corner of the projectile, where the plug begins to form.

6.2 METHOD OF EVOLVING THE PLUG

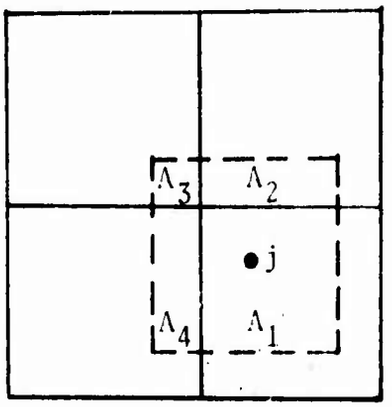
To simulate plugging failure, the HELP code essentially evolves a plug package the shape and mass of which are determined by a plastic work failure criterion and by the associated angle of maximum shearing stress.

6.2.1 Plugging Failure Criterion

The failure criterion used to propagate the vertical edge of the plug is based on the specific plastic work of the target material in the region near the endpoint of the plug edge. The plastic work of this material is approximated by using passive tracer points that move with the target material and by incrementing, by an area weighted average, the plastic work of the material associated with each point. Also associated with each point on a given cycle is an area weighted average angle of maximum shearing stress. Once the plastic work associated with one or more points in the region of the endpoint exceeds a predetermined cutoff value (PLWMIN), the average, current angle of maximum shearing stress associated with those points is used to determine the direction in which the vertical edge of the plug is extended.

Each cycle in SPHASE, a plastic work increment and an angle of maximum shearing stress are associated with the center of cells in the target. An area weighted average of these cell-centered quantities is used to define an angle of maximum shearing stress and to update the total plastic work

for the material associated with each moving point. For example, consider point j in the figure below.



The plastic work, W, associated with point j at time n is defined as

$$W_j^n = W_j^{n-1} + \sum_{i=1}^{4} A_i \cdot \Delta W_i^n / \sum_{i=1}^{4} A_i \quad . \tag{6.1}$$

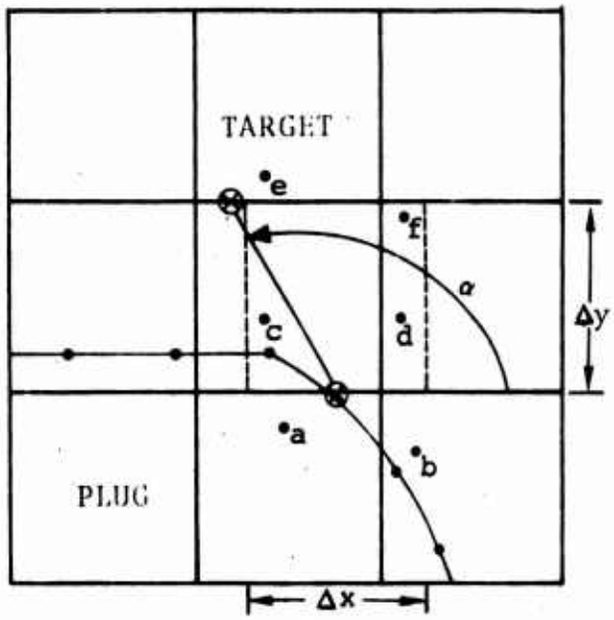
The average angle of maximum shearing stress, α, for the point is defined as:

$$\alpha_j^n = \sum_{i=1}^{4} A_i \cdot \alpha_i^n / \sum_{i=1}^{4} A_i \quad . \tag{6.2}$$

In both cases A_i denotes the overlap area of the pseudo-cell, centered at point j, with cell i. ΔW_i^n and α_i^n are the time n plastic work increment and angle of maximum shearing stress, respectively, associated with the center of cell i.

Each cycle, after the plastic work associated with the moving points has been updated, the code considers a region

which, as illustrated below, is one cell in area and lies above the intercept between the topmost horizontal grid line and the plug's vertical surface. The radial position of the region is centered about this intercept.



If one or more of the moving points which lie within that region has an accumulated plastic work total greater than the specified empirical value, PLWMIN, then the vertical edge of the plug is extended to the next horizontal grid line. The angle at which the plug edge is extended is an average of the current angles of maximum shearing stress associated with those points within the specified region that have exceeded the plastic work cutoff.

For example, in the case illustrated above, the code would consider the plastic work associated with passive tracer particles c, d, and f, only, since passive tracers a, b and e are outside the specified region. If, on a given cycle, the plastic work associated with all three points c, d and f exceeds the specified value, PLWMIN, the angle, α, at which the plug will be extended will be $\alpha = \frac{1}{3}(\alpha_c + \alpha_d + \alpha_f)$. If only point c satisfied the failure criterion, then $\alpha = \alpha_c$.

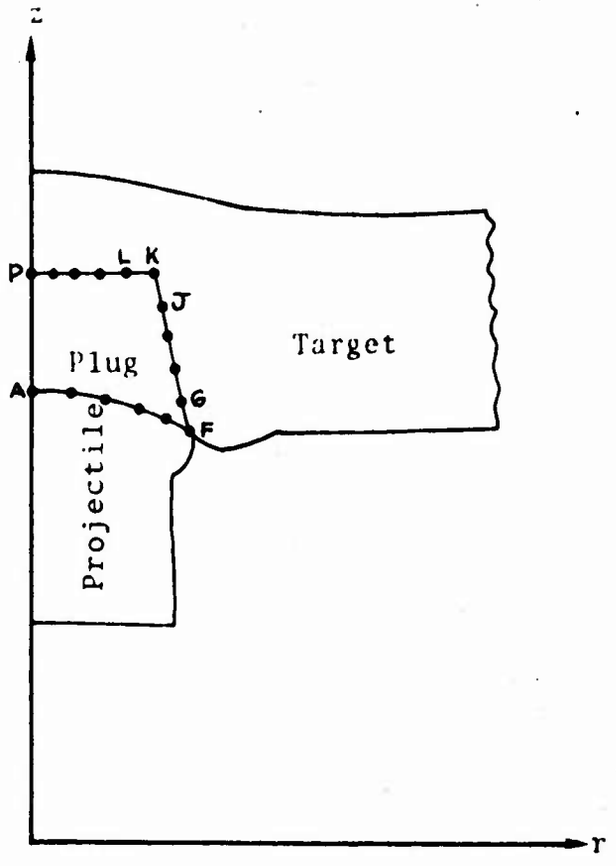
When the criterion for extending the plug surface is met, the plug package itself must be expanded. The mechanisms for this expansion are described in the following sections.

6.2.2 Conventions Governing Plug Tracer Particles

There are three stages of the plug formation. The conventions governing the initial placement of the plug tracer particles have been given in Section 6.2.1. The conventions for the intermediate and final stages are given below.

6.2.2.1 Partially Formed Plug

Once the plug edge is extended, the projectile, target, and plug tracer particles follow the conventions illustrated by the example below, in which 5 of the set of n duplicate plug package tracer particles generated initially by PLGADD (see Section 6.1.2) now define the vertical edge of the plug.



Projectile - The last six projectile tracers (excluding the dummy tracer) are at positions F through A.

Target - The first six target tracers are at P through K. The next (n-5) are also at K, where n is the number of duplicate points initially specified by NBGSD(2) and NENDSD(2). The next five are at J through F.

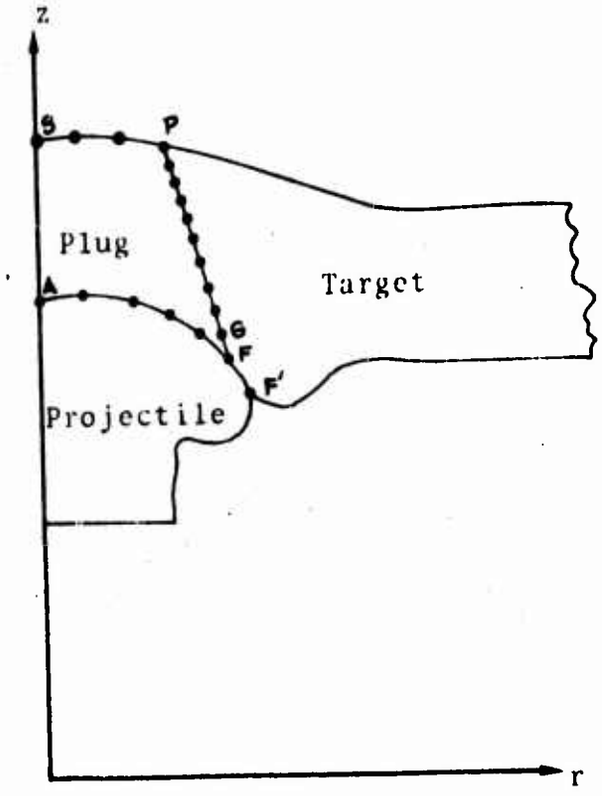
Plug - The first six plug tracers are at A through F. The next four are at G through J. The next (n-5) are at K, and the last five (excluding the dummy tracer) are at L through P.

Each time the plug is extended all but one of the duplicate particles is moved to the new endpoint of the vertical edge of the plug. The one particle left behind helps to define the vertical plug interface.

It should be noted here that the interface $\overline{PK}$ in the example above does not represent a discontinuity in the material as does the slipline interface $\overline{KF}$. Since $\overline{PK}$ is a psuedo-interface, the material flow field is unaffected by its presence.

6.2.2.2 Completely Formed Plug

Once the plug edge is extended to the free surface back of the target and the projectile and target lips coincide, the projectile, target and plug tracer particles follow the conventions outlined in the example below.



Projectile - The last six projectile tracers (excluding the dummy tracer) are at positions F through A. The target and projectile have joined between F and F´ and the slipline extends from P to F´.

Target - The target package no longer connects with the axis. The first eleven tracers are at P through F. The last target tracer (excluding the dummy tracer) is also at P in order to completely circumscribe the target package.

Plug - The first six plug tracers are at A through F. The next ten are at G through P, and the last four (excluding the dummy tracer) are at P through S. Note that there are two duplicate plug points at P; one is moved with plug velocities and defines the

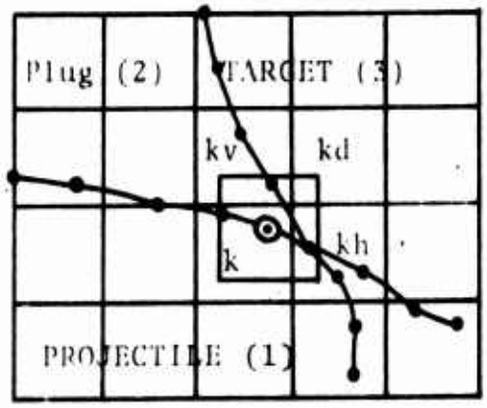
top right corner of the plug as it is pushed out; the other is moved with target velocities and defines the top left corner of the target. The last three tracers of the plug are coincident with what were the last three tracers of the target just before the plug was completely formed.

6.2.3 Movement of Plug Tracer Particles

As described in Section 4.1.2, tracer particles are moved with a density and area-weighted average velocity. However, to better simulate plugging and penetration mechanisms this method has been specialized for those plug, projectile and target tracer particles that define the plug package once it has begun to evolve.

1. The tracer particles along the projectile-plug interface are not moved with target material velocities, only plug and projectile velocities.
2. The tracers along the slipline edge of the plug, except for the top endpoint, are also moved with only plug and projectile velocities.
3. The tracers along the top surface of the plug, except the top endpoint, are moved normally using all velocities in the overlap region.
4. The top endpoint of the plug surface is moved with only target velocities.

The following example illustrates the way in which a tracer in category 1 above is moved. If the area associated



with one of these tracers overlaps a cell which contains only target material, e.g., cell kd in the figure, the overlapped area and its associated velocity component are ignored in the definition of the tracer's velocity components. The axial velocity of the circled point, therefore, is

$$v_p = \left(A_k \cdot (vs_1 + vs_2)/2 + A_{kv} \cdot vs_2 \right.$$

$$\left. + A_{kh} \cdot (vs_1 + vs_2)/2\right)/(A_k + A_{kv} + A_{kh}) \qquad (6.3)$$

where A_i is the overlap area between cell i and the pseudo-cell, and vs_j is the axial velocity of material j in cell i. Notice that the target velocity, vs_3, in the cells was not included in the average. The cell centered velocities are used in the averages only when the overlap cell is pure. The reasoning behind these special procedures is that the slipline represents a discontinuity in the velocity field and a tracer on one side of this line should not be affected by the motion of the material on the other side.

The tracers along the free surface of the target are moved with target velocities only. Tracers along the free surface of the plug are moved with plug velocities only. The

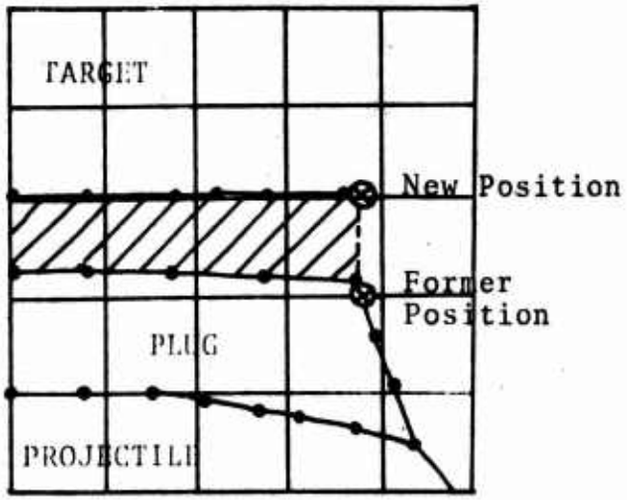
tracers along the free surface of the projectile are moved with both plug and projectile velocities in the case where the pseudo cell overlaps an interface cell which contains plug material. Otherwise they are moved only with projectile material velocities.

Once the plug surface has extended to the rear free surface of the target, the top endpoint of the slipline is considered a part of the target free surface; therefore, it is moved with target material velocities only. As the plug is pushed out, the slipline tracers below the top endpoint will, in general, have a larger axial component of velocity than the endpoint since they are moved with plug and projectile material velocities. The plug slipline tracers are accumulated at the endpoint. The corresponding target slipline tracers are removed and the number of target tracers is reduced accordingly.

6.2.4 Conversion of Target Material into Plug Material

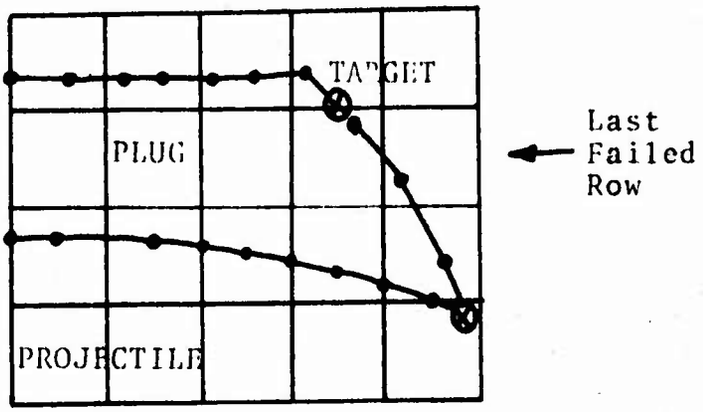
6.2.4.1 Redefinition of Tracer Particles

When the plug surface is extended, the top edge of the plug is redefined, and the plug package boundary encompasses a larger volume as illustrated below.

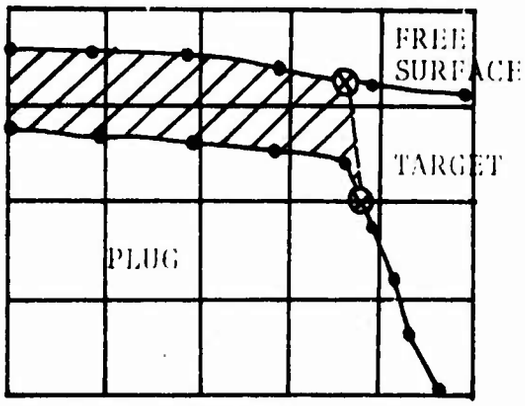


The tracers at the top of the plug are moved a fraction above the next horizontal grid line, which is also the new y-coordinate of the slipline endpoint. However, as illustrated below, those tracers at the top edge of the plug that are already above the new slipline endpoint are not moved.

Until the plug is completely formed, the upper endpoint of the slipline is on the top horizontal grid line of the last failed row. However, the top and vertical edges of the plug may extend above this horizontal line, as shown below, where ● denotes the slipline endpoints.



However, when the plug edge is extended into a cell that contains the free surface, the rear of the target has been reached, and the plug edge is extended beyond the grid line until it intersects the free surface, as illustrated below.

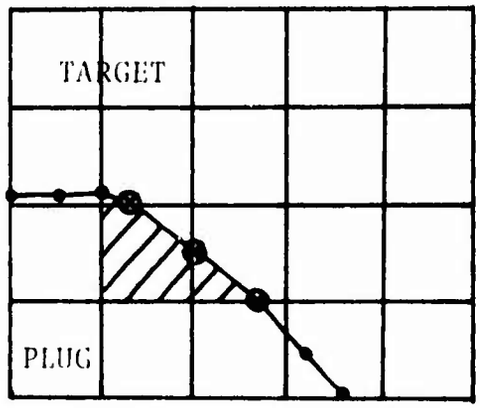


The tracers at the top of the plug are not moved in this case. Instead, the target tracers to the left of the new plug surface endpoint replace the tracers at the top edge of the plug. At this time in the calculation the number of plug and target tracers is reduced. All but two of the repeated plug tracers at the plug surface endpoint are removed. One of these will define the top right corner of the plug as it is pushed out; the other will define the top left corner of the target and will also be the endpoint of the slipline. The target package is no longer attached to the axis, and the first and last target tracers define the new slipline endpoint.

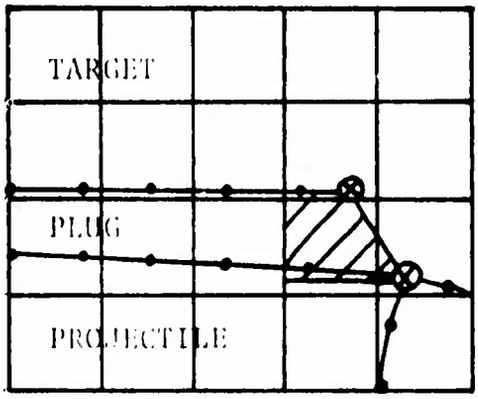
Accordingly, two free surface tracers are added at the slipline endpoint; one will move with the plug, the other will move with the target and remain at the slipline endpoint.

6.2.4.2 Redefinition of Target and Plug Masses

When the plug edge is extended and the plug package boundary is enlarged, the code converts some of the target material into plug material. To do this, the code computes the volume to be associated with plug material in cells containing the plug edge extension. Using the intercepts of the plug edge extension with the cell boundaries, this volume is computed, as denoted by the shaded regions in the figure below.

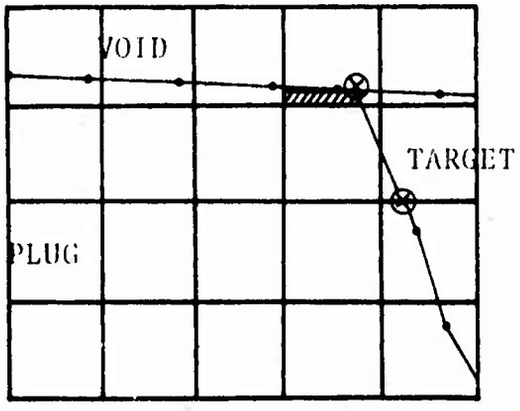


The initial and final extensions of the plug edge are special cases. The initial extension usually crosses one or more cells that contain projectile as well as target material. In this case, any curvature of the top of the projectile is ignored and the entire top surface of the projectile is assumed to have the same y-coordinate as the beginning point of the plug surface. The partial volume associated with the plug, therefore, is measured from the approximate position of the top of the projectile rather than from the bottom cell boundary, as illustrated below.



The partial volume associated with the target is defined as the total cell volume minus the plug and projectile partial volumes, where the projectile partial volume is based on its mass and density.

In the second special case, when the plug edge is extended to the free surface, the new top boundary of the plug is the top of the target rather than a grid line. Ignoring the small curvature of the target, the partial volume associated with the plug in the cell containing the endpoint is based on the y-coordinate of the new endpoint of the slipline, as illustrated below.



Once the volumes associated with the plug material are computed, the target material in the region of the plug extension is converted into plug material. The pure cells to the left of the plug surface endpoint are converted simply by changing the value of MFLAG for those cells from 3 to 2. (Package 3 is the target; package 2 is the plug.)

The XMASS, SIE, and RHO arrays are redefined for the interface cells that contain both plug and target material. Given m = MFLAG(k) - 100, the plug mass is stored in XMASS (2,m) and is defined as the product of the volume of the plug material for that cell and the average material density for that cell.

$$XMASS(2,m) = V_{plug} \cdot \rho_{av} \qquad (6.4)$$

where

$$\rho_{av} = m_{cell}/V_{cell} \quad . \qquad (6.5)$$

The target material is stored in XMASS(3,m) and is the difference of the total cell mass and the plug mass:

$$XMASS(3,m) = AMX(k) - XMASS(2,m) \quad . (6.6)$$

The density of both the plug and target materials is defined to be ρ_{av}, i.e., RHO(2,m) = RHO(3,m) = ρ_{av}. The specific internal energy of both materials is defined to be the average specific internal energy for the cell: SIE(2,m) = SIE(3,m) = AIX(k).

The procedure for an interface cell that will contain all three material packages varies from that described above. In this case, the volume of the target material is used to compute its mass, so that

$$XMASS(3,m) = V_{target} \cdot \rho_{target} \qquad (6.7)$$

where $V_{target} = V_{cell} - V_{plug} - V_{projectile}$. The density of the target material is used to define the plug mass:

$$XMASS(2,m) = V_{plug} \cdot \rho_{target} \quad . \qquad (6.8)$$

This follows from the fact that the plug is simply a part of the target. Therefore the density and specific internal energy of the plug material is defined to be the same as the density and specific internal energy of the target material:

$$RHO(2,m) = RHO(3,m)$$

$$SIE(2,m) = SIE(3,m) \quad .$$

All target material is converted to plug material in the interface cells to the left of the plug edge and at the free surface, i.e., the plug quantities are defined:

$$XMASS(2,m) = XMASS(3,m)$$

$$RHO(2,m) = RHO(3,m)$$

$$SIE(2,m) = SIE(3,m)$$

and then the target quantities are set to zero:

$$XMASS(3,m) = 0$$

$$RHO(3,m) = 0$$

$$SIE(3,m) = 0 \quad .$$

CHAPTER VII

DESCRIPTION OF INPUT VARIABLES AND RESTART PROCEDURES

7.1 GENERAL CAPABILITIES AND LIMITATIONS

A problem generator which uses the position of material interfaces to define cell quantities has been incorporated into the HELP code. Currently the generator includes a subroutine, TSETUP, which allows the user to easily generate material interfaces which are composed of straight line segments or portions of circles or ellipses. However, the method is applicable to any topological configuration, provided the user generates the tracer particles which define the material interfaces. The following general capabilities and limitations of the HELP code should be noted before applying it to a specific problem:

1. The number of material packages in a calculation is limited only by the dimensions of the material arrays, which can be increased up to the limit of the user's computer core storage. (See Appendix A for dimensioning the arrays.)
2. A single cell can contain any number of material interfaces and any number of distinct materials.
3. Two material packages can actually contain the same material (use the same equation of state) but have different initial conditions (velocities, energies and/or densities).
4. A package can be divided into any number of spatially disconnected subpackages. (See instructions for defining material tracer particles in Section 7.2.5.)

5. The material for each package must be homogeneous; i.e., each pure cell within a package must have the same density, velocity and internal energy. This restriction can be removed by appropriate user-supplied statements in FILGRD.
6. The grid boundaries cannot serve as free surfaces because the code assumes, for example, that a package which extends to the right edge of the grid is infinite in the x-direction.
7. The grid must have at least three rows and/or three columns.
8. The DATA statements in COMDIM and EQST can be redefined if the user wants to model a material which is not in the EQST list of materials modeled by HELP. An inert material must be assigned a material number from 1 to 19, inclusively. A gas or high explosive must be assigned a material number from 20 to 30, inclusively. In order for the pressure iteration to converge, any equation of state which is used in place of the Tillotson, the γ-law or the JWL, must be continuous and have a negative slope everywhere in the P-V plane, or at least prevent the iteration from entering any region which violates these restrictions. (See Section 4.6.1.3.)
9. A calculation can be made in plane or in cylindrical coordinates. (See definition of IGM in Section 7.2.1.)
10. The top and right grid boundaries are transmittive. The left boundary is reflective (an axis of symmetry in cylindrical coordinates).

The bottom boundary is the only one which can be either transmittive or reflective (see definition of CVIS in Section 7.2.1).

7.2 INSTRUCTIONS FOR GENERATING A PROBLEM

Instructions for generating a HELP problem are given in the sections that follow. An abbreviated set of input instructions on keypunch forms is given in Appendix B.

7.2.1 Z-Block Variables

Most of the options, flags, and problem parameters (such as the number of rows and columns in the grid) are variables located within the first 150 words of blank common and are defined by the first set of input cards. These variables are called Z-block variables because they all follow Z(1), the first word of blank common, and can be referenced using an appropriate subscript, between 1 and 150, for Z. For example, CVIS and Z(27) have the same location in blank common. Not all variables in the Z-block are defined by input cards; some are defined and updated by the code as the calculation proceeds. All Z-block variables are written on the restart file, and most are parameters which must be saved in order to restart a calculation. However, the variables such as UN93 or NUN32, which correspond to Z(93) and Z(32), respectively, are not currently being used by the code and are available to the user to define additional problem parameters as the need arises.

The Z-block variables are defined by cards that are read by subroutine CARDS. The format (I1, I5, I1, 7E9.4) is used to read the variables IEND, LOC, NUMWPC, (CARD(I), I = I, NUMWPC) which are defined as below:

IEND IEND=0 indicates that the variable(s) defined by this card are <u>all</u> typed real.

IEND=2 indicates that the variable(s) defined by this card are all typed integer.

IEND=1 indicates that the variable(s) defined by this card are all typed real, and this is the last card CARDS will read until it is called again by INPUT. (Note: a setup deck has three cards where IEND=1, a restart deck has only two such cards. When generating a problem, INPUT calls CARDS three times; when restarting a problem, INPUT calls CARDS twice.)

LOC LOC is the location in blank common of the first variable being defined by this card.

NUMWPC NUMWPC is the number of consecutively stored Z-block variables of the same type being defined by this card.

CARD(I) The value assigned to the I^{th} variable. NOTE: even if the variable is typed integer, the value must be entered with a decimal point. (See sample input decks in Chapter XI.)

The Z-block variables defined by input cards are listed and defined on the following pages. The type (integer or real) and location in blank common are indicated for each variable. The code assigns non-zero default values to some of these variables at the beginning of subroutine INPUT. The non-zero default values are given in parentheses after the variable definition. If no value is indicated, the default value is assumed to be zero.

INPUT Z-BLOCK VARIABLES

Variable Name	Column One Flag	Location in Blank Common	Definition (Default Values are in Parentheses)
PK(1)	1	151	Problem number. Any number from .0001 to 99.9999. It must be identical to PROB.
PROB		1	Problem number. (Same value as PK(1).) Must be included in the setup deck.
NFRELP	2	5	Gives frequency of "long" EDIT prints, which give velocities, energy, compression (or density) and stresses for all cells in active grid. A "short" EDIT gives this information only for the first column of cells, e.g., when NFRELP=2, every second EDIT print is "long," when NFRELP=5, every fifth EDIT print is "long", etc.
NDUMP7	2	6	Gives frequency of restart tape dumps relative to EDIT prints. (Program dumps only when EDIT prints.) E.g., if NDUMP7=5, a restart dump will be made every 5th time EDIT prints.
ICSTOP	2	7	Gives cycle at which execution will stop if the user wants to specify a cycle, rather than a value of T (time), on which to stop. If stopping on time omit this card.
NUMREZ	2	12	The total number of times the grid will be automatically rezoned. If not automatically rezoning the grid, omit this card.
KUNITR	2	14	The name of the file INPUT will read from. (7)

INPUT Z-BLOCK VARIABLES

Variable Name	Column One Flag	Location in Blank Common	Definition (Default Values are in Parentheses)
IPR	2	15	Maximum number of iterations CDT will perform when attempting to equilibrate the pressures of materials in a multi-material cell. If the pressures are not within an epsilon (PRCNT) of the average pressure after IPR iterations, an error exit occurs. (35)
PRCNT		16	Convergence criterion for the pressure equilibration of material in a multi-material cell. All material pressures, P_i, must satisfy the following: $\lvert(P_{av} - P_i)/P_{av}\rvert <$ PRCNT. (.001)
KUNITW	2	17	The name of the file EDIT and SETUP will write on. (7)
IGM	2	21	When IGM = 0 code uses cylindrical coordinates. When IGM = 1, code uses plane coordinates.
DMIN		24	The maximum relative error in the energy sum that the user wants to tolerate. The error is computed as follows: $\left(\sum_{k=2}^{KMAX} E_k - ETH\right)/ETH$ where E_k is total energy of cell k and ETH^k is theoretical energy in the grid--computed in SETUP and updated in HPHASE, TPHASE, SPHASE, ADDENG, REZONE, and RNDOFF. (10^{-3})
CVIS		27	A flag that describes the condition of the bottom grid boundary. If CVIS = 0, the bottom boundary will be reflective. If CVIS = - 1, the bottom boundary will be transmittive.

INPUT Z-BLOCK VARIABLES

Variable Name	Column One Flag	Location in Blank Common	Definition (Default Values are in Parentheses)
IMAX	2	33	The number of columns in the grid. If planning to rezone the grid in the x-direction, IMAX must be an even number. IMAX also must be at least 3.
JMAX	2	35	The number of rows in the grid. If planning to rezone in the y-direction, JMAX must be an even number. JMAX also must be at least 3.
MAPS	2	42	When MAPS > 0, part of the EDIT print is a set of symbolic maps of the compression (or density), pressure, radial and axial velocities, and specific internal energy of the cells in the active grid. MAPS=1 gives a compression map. MAPS=2 gives a density map.
NUMSCA	2	43	The number of times the cycle or time interval between EDIT prints is increased. (See PRLIM)
PRLIM		44	The time or cycle at which the EDIT print frequency is changed. Example: you wish to print every 10^{-8} sec until T = 10^{-7} sec; and every 10^{-7} sec until T = 10^{-6} sec, and every 10^{-6} sec thereafter. Set: PRDELT = 10^{-8} IPCYCL = 0. PRLIM = 10^{-7} PRFACT = 10. NUMSCA = 2.

INPUT Z-BLOCK VARIABLES

Variable Name	Column One Flag	Location in Blank Common	Definition (Default Values are in Parentheses)
			NOTE: When PRDELT is increased by a factor, PRLIM is also increased by the same factor for the next rescaling. If you want a constant print interval, omit NUMSCA, PRLIM, PRFACT.
PRDELT		45	The time (sec) between EDIT prints. If printing on cycle intervals, omit this card.
PRFACT		46	The factor by which the print interval (PRDELT or IPCYCL) and the print limit (PRLIM) are increased. (See PRLIM)
I1	2	47	The number of columns in the grid that have non-zero velocities or energy plus 2. I1 and I2 define the "active" grid. (Many quantities are computed only for cells inside the active grid.)
I2	2	48	The number of rows in the grid that have non-zero velocities or energy plus 2.
IPCYCL	2	49	The number of cycles between EDIT prints. If printing on time intervals, omit this card.
TSTOP	1	50	The value of T (time) at which the calculation will stop. When stopping on a specified cycle (ICSTOP > 0), set TSTOP = 0. NOTE: This card has a "1" in column one. It must always be included in both the setup deck and the restart deck.
IPLGRT	2	56	The rightmost column of the plugging region of the target. (2-3 columns to the right of the expected location of the plug's vertical edge.) Omit this card when not using the plugging option.

INPUT Z-BLOCK VARIABLES

Variable Name	Column One Flag	Location in Blank Common	Definition (Default Values are in Parentheses)
PLWMIN		59	The required minimum specific plastic work (ergs/gm) which the material near the top of the vertical edge of the plug must have before the plug edge is extended to the next row. See Section 6.2.1. Omit this card when not using the plugging option.
IPLGBT	2	60	The bottom-most row of the plugging region of the target. (Usually the first row of the target.) Omit this card when not using the plugging option.
IPLGTP	2	61	The top-most row of the plugging region of the target. (Usually the top row of the target.) Omit this card when not using the plugging option.
GAMMA		62	γ in $P=(\gamma-1)\rho E$ for material #20, an ideal gas.
NMAT	2	68	The number of material packages, excluding the void package.
CYCMX		69	The number of passes through subroutine INFACE each cycle to minimize transport noise near interfaces. (2)
CYCPH3		70	The number of passes through subroutine SPHASE (strength phase) each cycle. When the calculation is purely hydrodynamic (no strength effects), set CYCPH3=-1.(1)
NTRACR	2	72	The number of material tracers per cell diagonal. Used by ADDTCR when adding tracers. ADDTCR adds tracers only when the distance between two consecutive tracers in a specified region is greater than a cell diagonal divided by NTRACR. MINX, MAXX, MINY, MAXY specify the region of the grid considered by ADDTCR. NADD specifies the frequency with which ADDTCR is called.

INPUT Z-BLOCK VARIABLES

Variable Name	Column One Flag	Location in Blank Common	Definition (Default Values are in Parentheses)
NMXCLS	2	73	The maximum number of interface cells to exist in the grid on any one cycle during the calculation. This maximum should correspond to the dimensions of the material arrays (XMASS, SIE, etc.). See Appendix A.
NTPMX	2	78	The maximum number of tracer particles per material package. This maximum should correspond to the dimensions of the TX and TY arrays.
NTCC	2	81	When NTCC > 0, NTCC passive tracer particles are initially placed in the center of every fourth nonempty cell and subsequently moved through the grid with the material. These are used only for producing particle plots and do not affect the material behavior. When this card is omitted (NTCC = 0), cell centered tracers are not computed unless the plugging option is being used, in which case they are automatically generated in the plugging region of the target. (See Section 6.1.3)
SIEMIN		82	A cutoff on the total specific internal energy increment of a cell in TPHASE. This cutoff prevents small numerical signals from enlarging the active grid. (10^5)
EMIN		85	The minimum value of specific internal energy to be used in the ideal gas equation of state. (10^7)
PMIN		86	A pressure cutoff. If $\|P(k)\| <$ PMIN, then $P(k) = 0$. (5×10^6)

INPUT Z-BLOCK VARIABLES

Variable Name	Column One Flag	Location in Blank Common	Definition (Default Values are in Parentheses)
INTER	2	87	A special editing flag for debugging: =2 for debug prints in CDT =3 for debug prints by EDIT after HPHASE and SPHASE =5 for debug prints in TPHASE =7 for debug prints in SPHASE =11 for debug prints in UVMOD =13 for debug prints in UVCALC (See Section 9.2 if more than one debug is desired.)
REZ		95	The flag which initiates the rezoning of the grid. If NUMREZ > 0, this flag will be set automatically by the code. (See Section 8.4) However, the user can force a rezone on any restart cycle by setting REZ = 1 in the restart deck. (IEXTX and/or JEXTY must also be defined in order for the grid to be rezoned.)
NODUMP	2	96	When NODUMP = 1, EDIT will not write any restart dumps. (This flag overrides the NDUMP7 option, but does not prevent SETUP from writing the cycle 0 dump.)
NLINER	2	105	The package number of the shaped charge liner material that forms the jet. Used only when calculating the collapse of a liner.
NVRTEX	2	109	The second index of the void tracer particle that is at the vertex of the void closing region, i.e., (TX(NVOID, NVRTEX), TY(NVOID, NVRTEX)) are the coordinates of the vertex point. Used only when the code's automatic void closing routine is to be activated. (See Section 8.4)

INPUT Z-BLOCK VARIABLES

Variable Name	Column One Flag	Location in Blank Common	Definition (Default Values are in Parentheses)
ROEPS		110	A round-off epsilon used to define cutoffs in the calculation. (10^{-5})
PLGOPT		111	If the user is generating a plugging calculation, PLGOPT must be set equal to 1. The special plugging mechanisms are activated only if PLGOPT = 1. (See Chapter VI before using the plugging option.)
NSLD	2	112	The maximum number of cells the user expects will contain the slipline on any one cycle. This maximum should also depend on the dimensions of the slipline arrays (MSLD, etc.).
FINAL		113	The final value of the stability fraction used in determining the time step. See STAB. (4)
NOSLIP	2	115	If no sliplines are to be generated, set NOSLIP = 1, which causes the instructions and routines that affect only slipline cells to be skipped.
LVISC	2	116	A linear artificial viscosity term is added to cell boundary pressures if LVISC = 1. (See Section 2.2.2.5.)
NADD	2	118	A flag for automatically adding material tracer particles in the region specified by MINX, MAXX, MINY, MAXY. If NADD = 10 ADDTCR is called every cycle which is a multiple of 10, (e.g., cycle 20, 30, ... 250, 260, etc.). NOTE: NTRACR (=Z(72)) must also be defined before material tracers can be added by ADDTCR.
MINX	2	119	These parameters specify the left and right columns and the bottom and top rows, respectively, of the region in which ADDTCR will add tracer particles. These variables must be defined when NADD $\neq$ 0.
MAXX	2	120	
MINY	2	121	
MAXY	2	122	

INPUT Z-BLOCK VARIABLES

Variable Name	Column One Flag	Location in Blank Common	Definition (Default Values are in Parentheses)
IEXTX	2	123	A rezone flag. The grid will be rezoned in the x-direction only when IEXTX = 1. Omit this card if the grid is not to be rezoned, or if it is to be rezoned in the y-direction only.
JEXTY	2	124	A rezone flag. The grid will be rezoned in the y-direction only when JEXTY = 1. Omit this card if the grid is not to be rezoned, or if it is to be rezoned in the x-direction only.
STAB		139	The initial value of the stability fraction used in determining the time step. If FINAL = 0, STAB is constant. Otherwise, its value is doubled each cycle until it reaches the value of FINAL. (10^{-3})
DTMIN		144	The minimum value for the time step. The program gives an error exit if CDT calculates a $\Delta t <$ DTMIN. (10^{-11})
CRATIO		148	The compression of the material in adjacent cells is computed. If the ratio of these cells' compressions is greater than CRATIO, their pressures are inverse compression weighted and their velocities are compression weighted to define cell boundary values in HPHASE. (See Section 2.2.2.4.)(10^{4})
BBAR		149	A constant used in the calculation of the local sound speed of materials other than ideal gases and high explosives. $C = C_0 + BBAR \cdot \sqrt{\lvert P \rvert}$. (.5)
EMOB	1	150	Dummy end card of a setup deck. This card is read by CARDS after all other input cards have been read. This card must not be included in a restart deck. Set EMOB = 0 in the setup deck.

7.2.2 Cell Dimensions

The second set of input cards define the cell dimensions. The x and y dimensions of the cells can vary; however, the dimensions of contiguous cells should not vary by more than 10% except in parts of the grid well beyond the region of interest. Large aspect ratios (e.g., $\Delta X/\Delta Y > 2$) are also discouraged in significant regions of the grid.

The cards which define the DX, DY arrays are read by subroutine SETUP. A set of cards defining DX are read, then a set defining DY. Each card specifies up to four different DX or DY values. The information is read into the NNT and TEMP arrays: (NNT(L),L = 1,4), (TEMP(L),L = 1,4), where NNT(L) is the number of columns (or rows) that have a DX (or DY) dimension equal to TEMP(L). After the x-dimension (DX) of all columns has been defined, the next NNT is set equal to 999, which indicates that the entire DX array is defined. Then the DY array is similarly defined by the next group of cards. After the y-dimension (DY) of all rows has been defined, the next NNT is set equal to 999, indicating that all y-dimensions have been defined. An error exit will result if the user does not define exactly IMAX DX's and exactly JMAX DY's.

7.2.3 Initial Density, Velocity, and Specific Internal Energy of Each Material Package

In addition to the material tracer particle positions, the definition of material packages requires identification of MFLAG(2), the flag of the cell in the bottom left corner of the grid, and the material code number and initial conditions of each material package.

The card which follows those defining the cell dimensions defines MFLAG(2). The user should be able to predict from the placement of the interfaces in the grid whether cell k = 2 will be a pure cell, an interface cell, or a void cell at

t = 0. If it will be a pure cell of package n, then set MFLAG(2) = n. If it will be an interface cell, then set MFLAG(2) = -1. (In the latter case its value will be changed to be greater than 100 after the interfaces are processed by CALFRC and VOLFND). If it will be a void cell, then set MFLAG(2) = 0.

After MFLAG(2) is defined, a set of cards which defines the material code number and initial conditions of each material package is read. The first card in this set defines material package 1, the second card defines material package 2, etc. NMAT cards must therefore be read to define all the material packages.

The material code numbers for 24 materials are listed in EQST as well as in Tables 2.1 and 2.2 of this report. The material code numbers are stored in the MAT array and are used whenever the equation of state constants (ρ_0,A,B,γ, etc.) are referenced. These equation of state constants are defined in DATA statements in EQST and in COMDIM, the "included" element. For example, if package 3 is lead (material code number 10) then MAT(3) = 10, ESCAPA(10) is its bulk modulus, A, and RHOZ(10) is its normal density, ρ_0.

The density (g/cc), specific internal energy (ergs/g) and the radial and axial velocities (cm/sec) of <u>all</u> the material in each package are specified by these cards. The knowledgeable user can vary these quantities within the package by adding to subroutine FILGRD coding which would redefine the appropriate variables.

When defining the <u>initial</u> density, RHOIN, of a material, the user should be aware of the <u>normal</u> density, RHOZ, specified by the code for that material. (See Table 2.1 and 2.2.) The user can change the definition of ρ_0 for a given material by redefining the appropriate RHOZ variable in the

DATA statement in COMDIM, the "included" element. The user can also specify an initial density, RHOIN, which is different from a material's normal density, RHOZ.

See Appendix B for the exact format of the cards which define MFLAG(2) and the material code numbers and initial conditions of the material packages.

7.2.4 Strength Constants for Each Material Package

The shear yield strength and the tensile strength of each material package are defined by the next set of cards. The shear yield strength constants, Y_o, Y_1, Y_2, E_m, G, are all set to zero if the material has no yield strength. If, however, the material has no tensile strength, AMDM(n) is set to 1. The definition of the yield strength terms is given in Section 2.3.2.3. The tensile failure threshold is described in Section 2.3.3.

The setup deck must contain NMAT cards defining strength properties, even when none of the packages have strength and SPHASE is being bypassed.

See Appendix B for the exact format of the cards which define the strength constants.

7.2.5 Material Tracer Particles

A material package boundary is generated by dividing it into a series of straight lines, and/or arcs of circles and/or arcs of ellipses.

A set of 2 or 3 cards is read by subroutine TSETUP to define each one of these segments. The first of these cards specifies: (1) LTYPE, what kind of a segment to generate - a horizontal line, a vertical line, a diagonal line, an arc of a circle, or an arc of an ellipse; (2) MPN, which material package is being generated; and (3) NPTS, how many tracer particles to put along the segment. If the segment is a

straight line, only one other card, which defines the endpoints of the line in centimeters, is read. However, if the segment is an arc, two other cards are read which define the beginning and ending angles of the arc in radians and the location of its center, or foci, in centimeters.

When defining the last segment of a material package or subpackage* boundary, the user must assign a negative value to LTYPE. This signals TSETUP to generate a dummy tracer particle which marks the end of a tracer string for a subpackage or package. The coordinates of this dummy particle are (-1000, 0), and an error exit will result if this dummy particle is not the last particle in the set of tracers for a material package. After all segments of all packages, including the void package, have been generated, a flag card, setting LTYPE = 100, is read by TSETUP. This signals that all the material tracer particles have been defined.

As the segments are generated they must form a continuous boundary interrupted only by the grid boundaries. (It is unnecessary to generate tracers along the grid boundaries.) Furthermore, the segments need to be generated in an order such that the interior of the package lies on the left as one moves between any consecutive pair of tracers. This latter requirement also dictates which endpoint must be specified first on the input cards.

Since every segment of an interface is described by two identical strings of tracer particles (one for each material package lying on opposite sides of the interface),

*A material package can be divided into any number of spatially disconnected subpackages. The tracer string for each subpackage must end with a dummy tracer.

the number of tracers generated along a segment must be identical for both packages, the order of the two sets of tracers along that segment being exactly opposite.

A good rule to use in deciding how many tracers to place along a segment is to generate two per cell edge. In regions where great distortions are likely to occur, the tracers may be generated more densely, although activating ADDTCR (see definition of NADD in Z-block variables) helps to insure that the tracers will not become too sparse.

The user cannot generate more than NTPMX (a Z-block input variable) tracers for a given material package boundary; in that event TSETUP will print a message and stop. The definition of NTPMX should match the dimensions of the tracer particle arrays, TX, TY.

See Appendix B for the exact format of the cards which define the material tracer particles.

7.2.6 Slipline Endpoints

When sliplines are being generated, SETUP reads a set of cards which defines the slipline endpoints and specifies each material package as a master, a slave, or neither. (If there are no sliplines in the calculation and if NOSLIP (a Z-block input variable) is set equal to 1, the slipline cards will not be read by SETUP and must be omitted from the input deck.)

One card is read for each package; the first card defines the first package, the second card defines the second package, etc. Each of the first two variables, MASTRD, NSLAVD, is set either to the package number or to zero, depending on whether the package is a master or slave. Each of the next two variables NBGMD, NBGSD is set either to the index of the first tracer of the slipline or to zero,

again depending on whether the package is a master or slave. Likewise, each of the last two variables, NENDMD, NENDSD, is set either to the index of the last tracer of the slipline or to zero, depending on whether the package is a master or slave.

If no segment of a material package boundary is a slipline then all of the variables above are set to zero for that package, and the package is neither a slave nor a master. It does not matter which packages are the masters and which are the slaves. Several packages can be slaves and several can be masters. In a cell that contains more than one slave package the code forces all slave packages to have the same velocity components; likewise, all master packages in a given cell have the same velocity components.

Slipline endpoints can be redefined when the calculation is restarted. The slipline cards are read immediately following the Z-block variables in a restart deck, provided NOSLIP = 0. Otherwise they are not included in the restart deck.

See Appendix B for the exact format of the cards which define the slipline endpoints.

7.2.7 Definition of High Explosive Detonation Points

SETUP reads two sets of cards in order to define the primary and secondary initiation points of one or more high explosive packages. (See Section 2.3.4.) The first set of cards defines the centimeter coordinates of the primary and secondary initiation points as well as the time delay (which may be zero) for detonating the primary detonation points. The cards which define the secondary initiation points must be ordered by the time to detonate each point

starting with the point having the minimum detonation time. There is one card for each detonation point, plus an end flag card.

The second set of cards defines the approximate region of detonation associated with each detonation point. These regions can overlap. There is one card in this set for each detonation point, but no end flag card.

If the calculation does not involve a high explosive, one blank card must be inserted in the deck.

See Appendix B for the exact format of the cards which define the detonation points.

7.3 RESTART PROCEDURES

During the calculation, HELP periodically writes on a tape or drum file the problem parameters and the current state of the material in each cell. By reading this file and a few input cards, the code can "restart" and continue a calculation from an intermediate point.

In general, a restart deck consists of five parts:

1. the heading card
2. the "restart" card
3. cards redefining various Z-block variables (if desired)
4. the "end of data" card
5. cards defining slipline endpoints (if the slipline option is activated; i.e., if NOSLIP = 0).

The heading card can contain any alphanumeric symbols between columns 2 and 72. The "restart" card defines three variables in the PK array which are described below. (Note: since the "restart" card contains three words, it must have a "3" in

column 7. See Section 7.2.1.) Also, since it is the only card processed by CARDS before the restart file is read, it must have a "1" in column 1. The third part of the restart deck allows the user to redefine any of the Z-block variables he desires; e.g. variables which make the code add tracer particles, change edit frequency, obtain debug prints, etc. If the user is stopping the calculation on cycles, he must define a new stop cycle, ICSTOP, and this is the only case in which there must be a card in this third part of the restart deck. The fourth part of the restart deck is the "end of data" card. This card, which must have a "1" in column one, redefines TSTOP, the value of time at which the current run will stop. If the calculation is stopping on cycles, rather than time, a zero value is entered. The last part of the restart deck, which defines the slipline endpoints, is necessary only if the slipline capability is being used in the calculation. (See Section 7.2.6 for discussion of defining the slipline endpoints.) As noted in Section 8.3, the slipline can be completely redefined or even removed during a restart. The user should be particularly aware of the possible change of endpoint indices if ADDTCR (see Section 8.2) and/or VDCLOS (see Section 8.4) have been called during the previous run. In general, unless the user wishes to redefine the endpoints of the slipline, he should input those indices listed in the EDIT print of the cycle from which the calculation is being restarted.

The restart deck of 6 cards listed below could be used to restart at cycle 78 a calculation whose problem number is 26.5 and to run it until T = 3.0 μsec. A "short" EDIT print of the restart cycle is indicated by PK(3) = −2. The third card redefines NADD, which controls the frequency

of the calls to subroutine ADDTCR (let's assume that MINX, MAXX, MINY, MAXY were defined previously). Until NADD is redefined again ADDTCR will be called every cycle which is a multiple of 10. The slipline cards indicate package 1 is a slave and package 2 is a master; 23 tracers in each package define the slipline.

```
        |  8-16  |  17-25  |  26-34  |  35-43  |  44-52  |  53-61  |  62-70  |
12345678901234567890123456789012345678901234567890123456789012345678901234567890

                    IRON - ALUMINUM IMPACT
1  151326.5      78.       -2.
2  11811.
1   5013.0    -06
    0    1    0   51    0   73
    2    0  167    0  189    0
```

Let's assume that this same problem was to be restarted on cycle 325, the slipline was to be removed and the calculation was to be stopped on cycle 326. The following 5 input cards would be used:

```
        |  8-16  |  17-25  |  26-34  |  35-43  |  44-52  |  53-61  |  62-70  |
12345678901234567890123456789012345678901234567890123456789012345678901234567890

                    IRON - ALUMINUM IMPACT
1  151326.5      325.      -2.
2  11511.
2    71326.
1   5013.
```

Card 3 sets NOSLIP = 1, which deactivates the slipline option. Thus it is not necessary to define the slipline endpoints in this restart deck.

It should be noted that any blank common variable that is not in the Z-block can be changed using the CARDS routine format if the user can compute its location in blank common. However, usually it is easier to code the changes into subroutine INPUT (following statement number 40) after the restart tape has been read and the pressure array has been initialized. Of course, this is the only way variables not in blank common can be redefined by the user.

7.4 REDEFINING TRACER PARTICLES WHEN RESTARTING A CALCULATION

In the course of some calculations, problems can arise because of the position of the material tracer particles. This is most likely to occur in calculations which involve severe velocity gradients or which employ VDCLOS, the automatic void closing routine (see Section 8.4). In the first case, the speed at which the tracer particles become sparse makes it easier for the tracers to cross each other, destroying the logic of FRACS. In the second case, it may happen, for example, that a tracer particle may pass to the left of the vertex point. This could create a situation in which VDCLOS forces the tracer string to cross itself when the tracers are moved to the new vertex position.

If these situations arise, it is necessary for the user to add, delete, or reposition some of the interface tracer particles on a restart cycle (probably the last one which was written before the problem arose) in order for the calculation to proceed smoothly. The procedure is not difficult, but must be carefully thought out before being implemented. Basically these are four principles which must be firmly adhered to:

1. Every interface tracer of each package, including the void, has a corresponding tracer from the other package(s) which adjoins that interface. In the case of a corner point among more than two packages,

such as the intersection of the liner, casing, and HE package in a shaped charge calculation, there may be more than one other corresponding tracer. If any tracers from one package are added, deleted, or moved to a new location, its corresponding twin(s) must also be added, deleted or moved to the identical new location.

2. If tracers are being added to or deleted from a material or void package n, the value of NMP(n), which stores the number of particles, including the dummy endpoint, defining the package n boundary, must be increased or decreased accordingly.

3. When adding or deleting tracers, the user must insure that the dummy endpoint (-1000, 0) is the value of the final tracer of each package and subpackage.

4. When adding or deleting tracers from calculations which use the automatic void closing routine, and/or the slipline option of the code, care must be taken to insure that the values of NVRTEX and/or the NBGMD, NENDMD, NBGSD, NENDSD arrays reflect the changes being made.

An example of a calculation which uses the automatic void closing routine will serve to illustrate the above procedures. Consider the situation in Figure 7.1, in which the package 2 tracer particles near the vertex of the void closing region have become too sparse to satisfy the conditions considered by VDCLOS, i.e., even if the pressure is positive in the cell containing the vertex point, the angle between the two surfaces is greater than .2 radians and is becoming larger. (See Section 8.4.2.) Therefore, this void must be closed by the user.

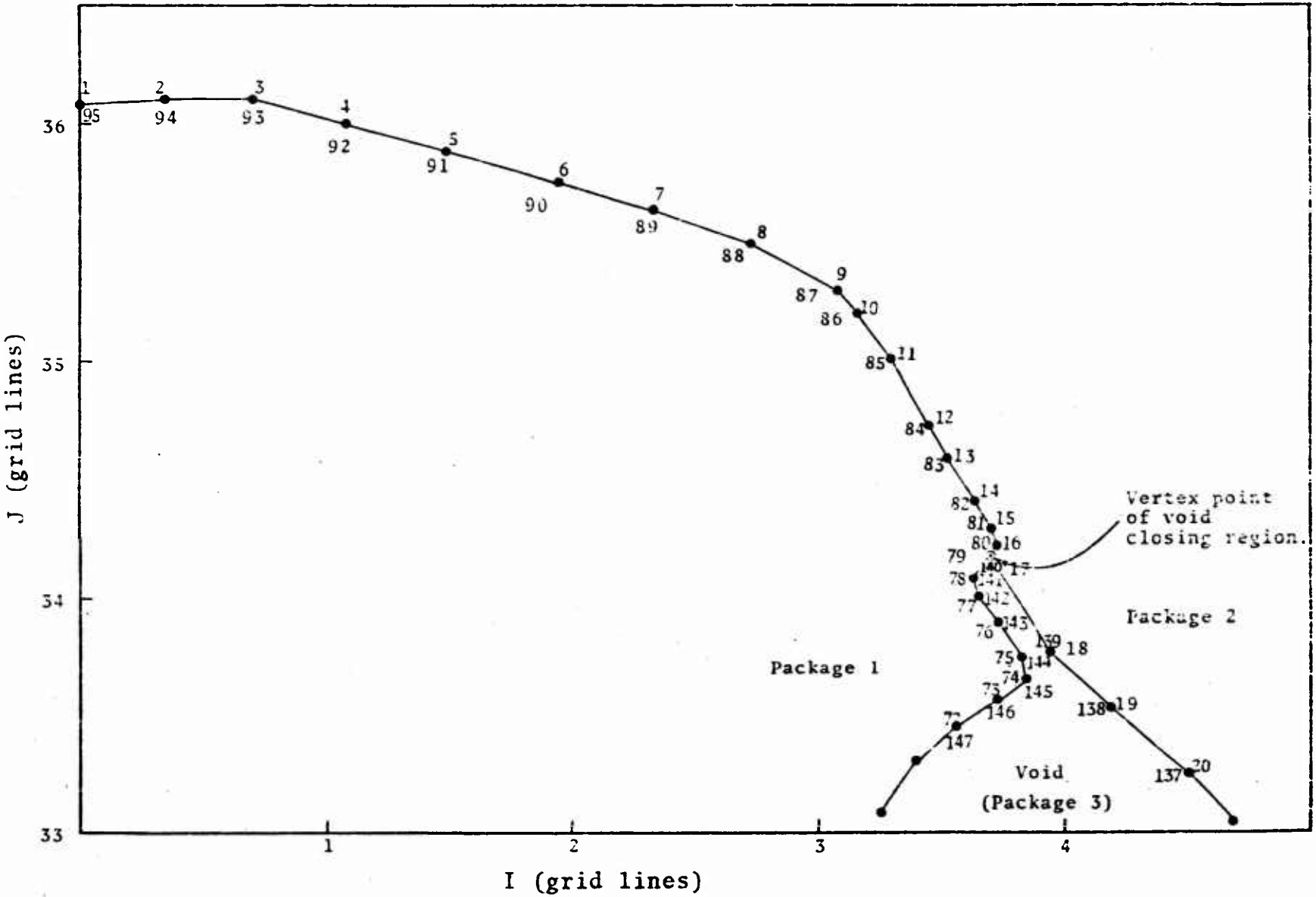


Figure 7.1--Tracer particle positions before being redefined by user.

In this example, the current vertex of the void closing region is defined by the 79th package 1 particle, the 17th package 2 particle and the 149th free surface particle (package 3). Let us assume the user decides the new vertex point will be defined by the 18th package 2 tracer. The 75th package 1 tracer will be moved to the new vertex point, while package 1 tracer particles 76 through 78 and free surface tracers 140 through 144 will be removed. Figure 7.2 illustrates the modified interface positions resulting from these changes to the tracer particles. Note that the 139th free surface tracer is now at the vertex, so NVRTEX must be set equal to 139 for VDCLOS to operate correctly.

These changes to the tracer particles can be made in subroutine INPUT as indicated by the following lines of FORTRAN:

```
C             *** INITIALIZE P-STORAGE.
   30 DO 40 K=1,KMAX
   40 P(K)=0.0
C
C             *** CHANGE COMMON VARIABLES HERE ON A RESTART
C
      IF(NC.GT.145) GO TO 355
C
      TX(1,75)=TX(2,18)
      TY(1,75)=TY(2,18)
C
      DO 351 L=79,96
      TX(1,L-3)=TX(1,L)
      TY(1,L-3)=TY(1,L)
  351 CONTINUE
      NMP(1)=93
C
      DO 352 L=145,219
      TX(3,L-5)=TX(3,L)
      TY(3,L-5)=TY(3,L)
  352 CONTINUE
      NMP(3)=214
C
      NVRTEX=139
C
  355 CONTINUE
```

lines added to INPUT to change tracer particle positions

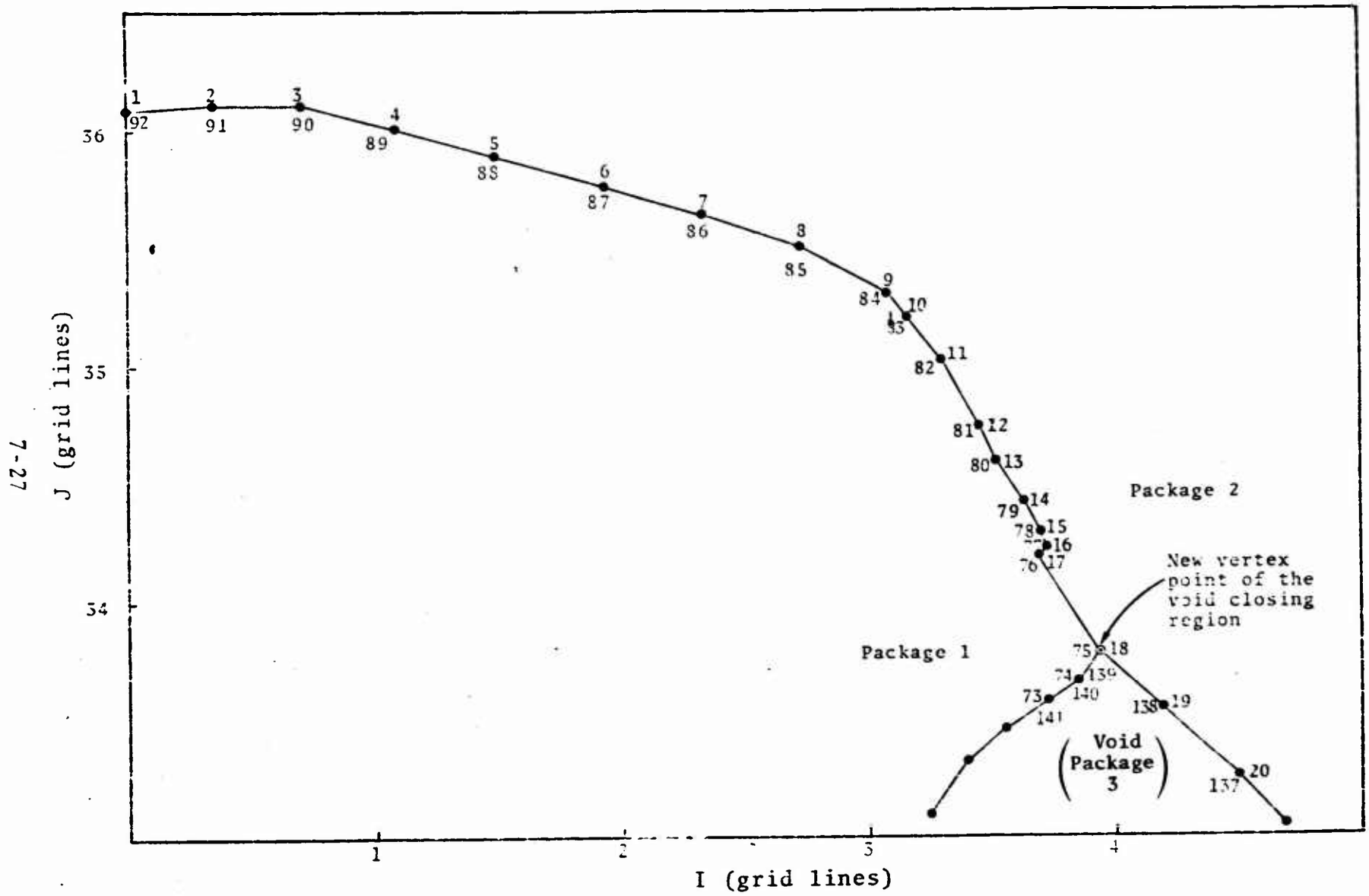


Figure 7.2--Tracer particle positions <u>after</u> being redefined by user.

In this case, the changes are being made after cycle 145 has been read off the restart file. The cycle 145 restart dump will have the unmodified positions (Figure 7.1) of the tracer particles; however, in the next cycle to be executed, number 146, INFACE will process the modified tracers as they appear in Figure 7.2, and the next restart dump will have the modified tracer positions.

CHAPTER VIII

INTERACTIVE CAPABILITIES

There are several ways the user can interact with the calculation as it progresses. The area encompassed by the grid can be enlarged and the resolution reduced by rezoning; the density of the tracer particles in a specified region can be increased; interfaces can become slip surfaces or slip surfaces can be removed; and a vertex of two intersecting free surfaces can be specified where the code will automatically "close a void." These interactive capabilities will be discussed in more detail in the sections that follow.

8.1 REZONING THE GRID

The HELP rezone reduces the resolution of the grid by combining cells and enlarging the area encompassed by the grid, keeping the number of cells in the grid constant. The HELP rezone combines two adjacent cells into one. Therefore, the grid can be rezoned radially and/or axially only if there is an even number of cells in the direction to be rezoned. The user should note that this rezone routine is not capable of rezoning plugging problems.

8.1.1 Combining Cells

When two cells, k and kn, are combined into one, the masses of the two cells are summed, and the velocity components (u',v') of the new cell are a mass weighted average of the velocity components of the original cells, thereby conserving mass and momentum.

$$m' = m(k) + m(kn)$$

$$u' = (u(k)m(k) + u(kn)m(kn))/(m(k) + m(kn))$$

$$v' = (v(k)m(k) + v(kn)m(kn))/(m(k) + m(kn)) \quad .$$

The material masses and velocities of interface cells are averaged in the same manner. For example, given m=MFLAG(k)-100 and mn=MFLAG(kn)-100, the mass of material n in the new cell, will be

$$m_n' = XMASS(n,m) + XMASS(n,mn) \quad .$$

The internal energy of the new cell is defined so as to conserve the total energy of the two cells being combined. Therefore, the internal energy of the new cell, IE', becomes the sum of the internal energies of the two cells plus the thermalized kinetic energy resulting from the definition of the new cell velocity components.

$$IE' = [IE(k)\cdot m(k) + IE(kn)\cdot m(kn) +$$

$$m(k)\cdot[(u(k)-u')^2 + (v(k)-v')^2] +$$

$$m(kn)\cdot[(u(kn)-u')^2 + (v(kn)-v')^2]]/(m(k) + m(kn)) \quad .$$

The internal energies of the materials in interface cells are redefined in the same manner.

The deviator stresses of the new cell are a mass weighted average of the deviator stresses of the two cells, whereas the slipline angle and the detonation time associated with the new cell are simple averages of those quantities in the two original cells.

8.1.2 Adding Material

The code automatically adds material above and/or to the right of the original grid, depending upon the direction

in which the grid is being rezoned. If, for example, the grid is rezoned in the x-direction, IMAX/2 cells are added to each row, since the combination of pairs of original cells has resulted in the original area being encompassed by half as many cells. The cells that are added on the right have the same x-dimension as the cell in the last column of the original grid. These added cells are also assumed to have the same initial properties (density, velocity and internal energy) that the material had in the initial problem setup.

If the last cell in a row is an interface cell, the code assumes that the interface extends throughout the row being added; therefore, all the added cells in that row will be interface cells. The user, in this case, should have generated tracer particles on a horizontal line beyond the area of the original grid. (Note: The rezone assumes that interfaces extending beyond the original grid are horizontal in the x-direction and vertical in the y-direction. Also, the rezone will not automatically generate a free-surface and a void unless they are an extension of a free surface and a void in the original grid.) Furthermore, the rezone routine, as it exists, does not add material on the left (in plane coordinates) nor below the grid. However, the knowledgeable user can remove these limitations quite easily.

8.1.3 Activating the Rezone

Basically the code implements two methods for rezoning the grid. One of these is an automatic rezone, in which the code automatically determines, during a run, when to rezone, does so, and continues the run to the desired final time or

cycle. The other method is a rezone which is triggered by the user through the restart deck. No matter which method is being used, the user must set the Z-block variables IEXTX and/or JEXTY to 1 to indicate the direction(s), radial and/or axial, respectively, in which the grid is to be rezoned. As noted before, the grid dimension(s) (IMAX and/or JMAX) in the direction(s) to be rezoned must be an even number.

8.1.3.1 Automatic Rezone

In order to activate the automatic rezone capability, the user must define the Z-block variable NUMREZ to be the maximum number of automatic rezones he wishes to allow. Subroutine TPHASE then tests to see if any mass is transported through the right grid boundary (if IEXTX = 1) and/or the top grid boundary (if JEXTY = 1). If any mass is lost through the boundary being tested, the rezone flag REZ is set to 1 and the code rezones on the next cycle.

The code automatically edits both before and after the rezone. Also the value of NUMREZ is decreased by 1 following each rezone so that a maximum of NUMREZ automatic rezones is allowed.

8.1.3.2 User-activated Rezone

The user has the option of triggering the rezone through the restart deck by defining the rezone flag REZ = 1. In this case the pickup cycle is rezoned in the directions indicated by the IEXTX and JEXTY flags. The code gives an edit following the rezone and continues the execution of the problem.

8.2 AUTOMATIC ADDITION OF MATERIAL TRACER PARTICLES

By defining the appropriate variables (NADD, NTRACR, MAXX, MINX, MAXY, MINY), the user can insure that the tracer particles will not become too sparse in the regions of the grid where there are great expansions or distortions of the material interfaces.

Every cycle which is a multiple of NADD, subroutine ADDTCR is called. ADDTCR searches the region bounded by the MINX, MAXX columns and the MINY, MAXY rows. If a pair of consecutive tracers in that region are more than 1/NTRACR of a cell diagonal apart, ADDTCR adds enough tracers between the pair to achieve the desired density of NTRACR tracers per cell diagonal. (Section 7.4 describes how tracers can be "manually" added by inserting coding changes in INPUT when a calculation is restarted.)

If, by adding a set of tracers, the dimension (NTPMX) of the tracer particle arrays would be exceeded, ADDTCR instead prints a warning and sets NADD to zero. The code will not call ADDTCR until NADD is redefined in the restart input deck, which, in this case, should be done only after the tracer particle arrays have been enlarged and the code recompiled. Also, ADDTCR redefines the slipline endpoint indices and the void closing vertex, NVRTEX, when necessary.

8.3 REDEFINING SLIPLINES

The sliplines in HELP can be lengthened, shortened, created or eliminated each time a calculation is restarted, simply by redefining the endpoints of the slipline. During the first cycle following a restart the code will reflect the change in the slipline definitions when it computes the value of THETA(m) for each interface cell.

An interface cell whose interface is changed from a slip to a non-slip surface, will have all material velocity components set equal to the cell centered velocity components and the resulting thermalized kinetic energy will be added to the material internal energies. This will be done automatically in TPHASE.

The material velocity components of interface cells that have changed from non-slip to slip cells, will gradually become different from the cell-centered values as the master and slave materials are treated differently in TPHASE and HPHASE.

It should be remembered (when redefining the endpoints of the slipline), that a slip cell has no shear yield strength.

8.4 CLOSING A VOID

The continuous velocity field used to move tracer particles in HELP prevents two particles on a collision course from ever actually meeting. As the particles come closer and closer to one another their velocity fields become more and more alike until they are being moved with almost identical velocity components. Therefore, if two free surfaces are moving toward one another, they will never quite meet, and the void will never quite close.

The void closing routine in HELP will automatically close the void in one specified region of the grid where two free surfaces initially have a common point; this common point is called the vertex in the discussion that follows.

8.4.1 Identifying Void Closing Region

The vertex point is identified by an input variable, NVRTEX, where (TX(NVOID, NVRTEX), TY(NVOID, NVRTEX)) are the coordinates of the void tracer which is at the common point. As the void is closed, NVRTEX is redefined, identifying another free surface particle as the vertex of the void closing region.

8.4.2 Criteria for Closing a Void

The void is closed if at least one of the following three criteria is met:

1. The pressure, PVRTEX, of the cell which contained the vertex particle at the start of the cycle is positive, and the angle between the two surfaces is less than .2 radians.
2. The two free surface interfaces have crossed. (This occurs only if the tracer particles have become too sparse in that region, or if the plugging option, which invokes special noncontinuous rules for moving tracer particles, is being used.)
3. The particle next to the vertex on one free surface is less than 1/10 of a cell diagonal from the other free surface.

8.4.3 Method for Closing the Void

VDCLOS defines four points which are used in testing the criteria listed above as well as in the void closing procedure. Referring to Figure 8.1, (TX1, TY1), (TX2, TY2), (TX3, TY3) are the void tracers indexed by NVRTEX-1, NVRTEX and NVRTEX+1, respectively (i.e., (TX2, TY2) = [TX(NVOID, NVRTEX), TY(NVOID, NVRTEX)]). The fourth point (TXP, TYP) is defined by VDCLOS

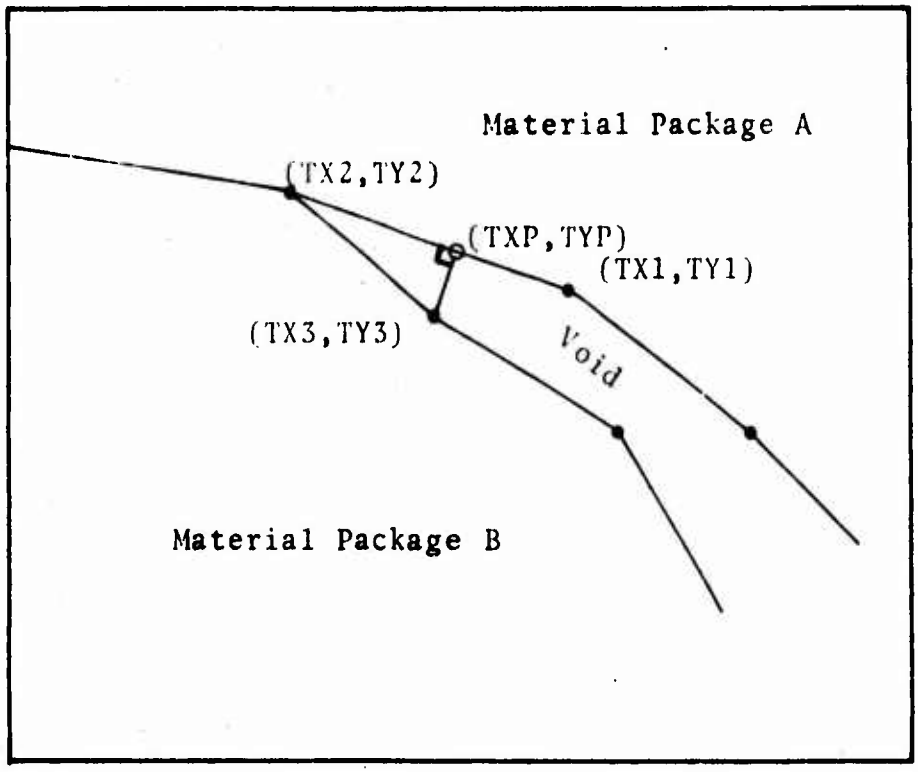


Figure 8.1--Void closing region before void is closed.

such that the line through (TX3, TY3) and (TXP, TYP) is perpendicular to the interface line through (TX1, TY1) and (TX2, TY2).

If one of the three criteria listed above is met, the following five steps are taken to close the void. These steps are described with reference to the example illustrated by Figures 8.1 and 8.2.

1. The package B tracer at (TX3, TY3) is moved to (TXP, TYP).
2. The package A tracer at (TX1, TY1) is moved to (TXP, TYP).
3. The void tracer at (TX1, TY1) is moved to (TXP, TYP).
4. The void tracers at (TX2, TY2) and (TX3, TY3) are removed from the string of void tracers, and NMP(NVOID) is decremented by 2.
5. Since the new vertex point is at (TXP, TYP), which is the location of the NVRTEX - 1 tracer (see step 3), NVRTEX is decremented by 1.

Each time the void is closed, the position of the vertex point is changed, and the void closing region is redefined. VDCLOS then applies the same three criteria* to the new void closing region, closing the void and redefining another void closing region if any of the criteria are met. This process continues until a void closing region is defined

---

* Since PVRTEX is the pressure of the cell containing the vertex point at the beginning of the cycle, the first criterion is not applicable if the new vertex point is not in that cell.

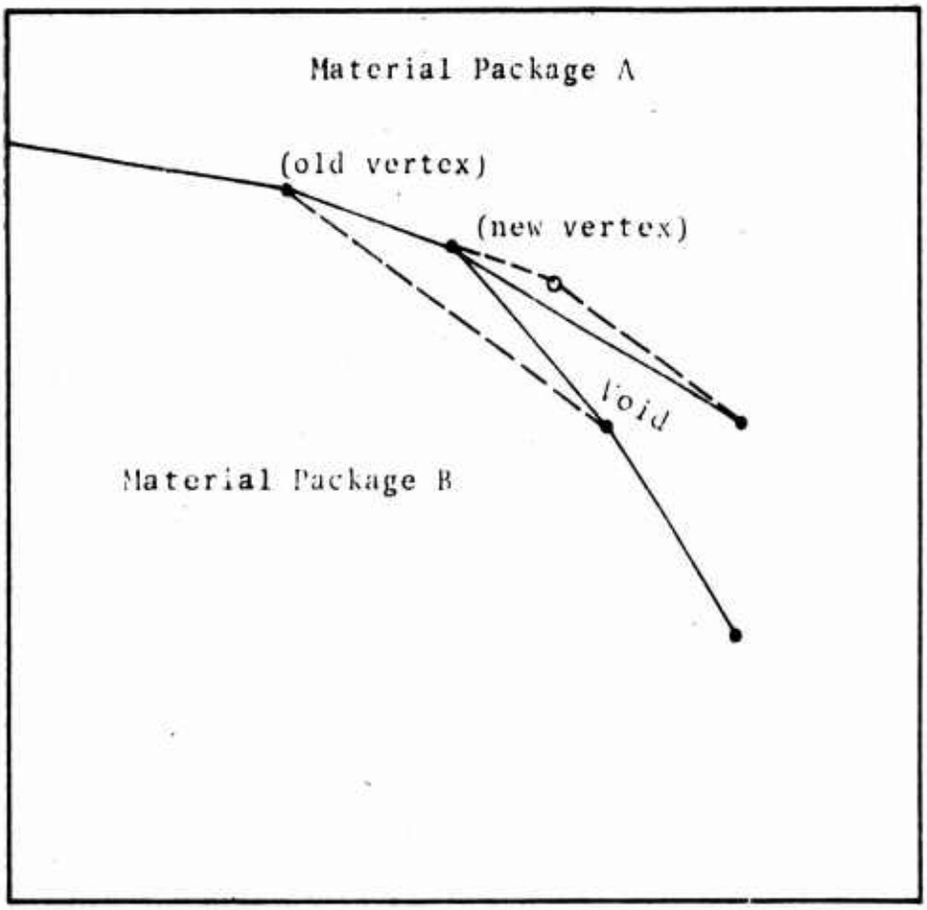


Figure 8.2-- Void closing region after void is closed. The old positions are represented by dashed lines, and the original position of (TX1, TY1) is represented by an open circle.

which satisfies none of the applicable void closing criteria. VDCLOS is then exited and not invoked again until the following subcycle of INFACE.

CHAPTER IX

ERROR CONDITIONS

The following sections should help the user diagnose a variety of error conditions which may occur during a HELP calculation. The error messages printed by the code are listed alphabetically in Section 9.1, along with the subroutines which print each message and comments which suggest what the user may do to rectify each problem. Section 9.2 describes the optional diagnostic prints the user may obtain in order to clarify an aspect of a calculation or an error condition.

9.1 ALPHABETIC LIST OF ERROR MESSAGES

The table that follows is an alphabetic list of the error messages printed by the HELP code. Each error is classified as fatal (F) or nonfatal (Nf); only fatal errors cause execution to stop. The subroutine line number of the statement which prints the error message is indicated in parentheses under the subroutine name. The comments direct the user to the most likely cause of the error condition; however, the user is encouraged to pursue the matter further, as there are undoubtedly situations which are not described here.

In many cases the user can correct the error by redefining an input parameter or by coding changes to cell quantities or tracer positions in subroutine INPUT. (See Section 7.4.) The calculation can then be restarted on the dump cycle before the error occurred unless the error occurred in one of the generator routines. However, if the error occurs at the beginning of a cycle, before SPHASE, HPHASE or TPHASE have

changed any of the cell quantities, and an error dump has been made as a result of a call to ERROR, the user can correct the error and restart from the error dump. (Error #32 is an example.)

	Error Message	Type	Source	Comment
1.	ALL SPECIFIC VOLUMES MULTIPLIED BY	Nf	CDT (147)	Normal iteration procedure not sufficient to conserve volume of cell.
2.	BLANK CARD	Nf	CARDS (26)	A blank card is in the set of cards read by CARDS.
3.	CHECK FIRST RECORD OF THE DUMP AND FIRST DATA CARD OF INPUT DECK	F	INPUT (199)	Indicates user is reading wrong tape or has asked for wrong cycle or problem number on tape.
4.	ERROR CONDITION IN THETAS M, MOS, (MSLD (N), N=1, MOS)	F	THETAS (78)	Flags (MSLD) defined by PTSAV are in error. Possibly a storage error.
5.	ERROR CONDITION - THETAS: THETA, XTP, XBT, YRT, YLF	F	THETAS (115)	Slipline does not cross this cell and yet PTSAV has indicated that it does. Possibly a storage error.
6.	ERROR EXIT - SEE STATEMENT NUMBER _____ IN _____	F	ERROR (9-36)	Refer to coding near the statement number in the indicated subroutine.
7.	ERROR IN ADDTCR. NUMBER OF TRACERS IN PACKAGE _____ EXCEEDED NTPMX.	Nf	ADDTCR (170)	To activate ADDTCR on subsequent run, change NTPMX and reset NADD. Tracer particle arrays may need to be enlarged.
8.	ERROR IN DETIME CELL I= J= HAS NO DETONATION TIME, K,N=	F	DETIME (131)	A detonation time has not been calculated for a cell containing explosive. Check detonation points.

	Error Message	Type	Source	Comment
9.	ERROR IN DETIME ROUTINE, SEARCHING FOR K= KW WAS INCREASED TO A VALUE .GT. KMAX OR .LT. 1	F	DETIME (425)	Indicates the search for the K-th cell is outside the grid. Can result from machine round off error, giving an erroneous value of I or J.
10.	ERROR IN INPUT CARD, IDET= INPUT CARD IS	F	DETIME (444)	Indicates the type of initiation point (IP) read is in error. Attempted to read primary IP data into the secondary IP array. Input cards are probably out of order.
11.	ERROR IN MFLAG DETERMINATION I= J= K=	F	FILGRD (134)	Usually a sign that MFLAG(2) was defined incorrectly in the input deck. May also be an error in material interface definition.
12.	ERROR IN PTSAV I,J, M1, M2, MT, THETA(MT)	F	PTSAV (35)	Slide line flag (MSLD) for cell I,J improperly defined. Probably a storage error.
13.	ERROR IN VOLUMES I= J= MFK= VSUM= VCELL=	Nf	FILGRD (31)	Sum of material partial volumes do not equal the cell's volume. Can probably ignore if error is very small and occurs for only one or two cells. Can be a result of interfaces being incorrectly defined by the input cards. Check especially that cards defining tracer particle positions for different materials adjoining a common interface segment have the same end points and define the same number of tracers.

	Error Message	Type	Source	Comment
14.	ERROR ON PRECEEDING DATA CARD	F	CARDS (28)	Probably numbers are punched in the wrong column on a card read by CARDS.
15.	EVACUATION OF MATERIAL N I,J,N,M,ML, MB TFLUX, XMASS(N,M), SAMMP(N.M), SAMPY(N,M)	Nf	DMADJ (126 145)	Not an error condition unless XMASS(N,M) is relatively large for evacuation into a neighbor cell. Can be caused by interfaces crossing; look for error 31.
16.	FRACS DETECTED AN INCOMPLETE SUBPACKAGE OF MATERIAL PACKAGE _____ CHECK REMAINING PACKAGES M1, M2, NN=, TX1, TY1, TX2, TY2 =	F	FRACS (562)	NMP(N) not properly defined or dummy endpoint not added when tracers were generated or manually changed.
17.	IMAX= WHICH IS NOT AN EVEN NUMBER. THE GRID WAS NOT REZONED IN THE X-DIRECTION	Nf	REZONE (27)	Grid must have an even number of columns to be rezoned in the X-direction.
18.	IMPROPER GRID DEFINITION IN SETUP. IMAX = I =	F	SETUP (149)	User did not define IMAX values of DX.
19.	IMPROPER GRID DEFINITION IN SETUP. JMAX = J =	F	SETUP (151)	User did not define JMAX values of DY.
20.	INSUFFICIENT AMOUNT OF MIXED CELL STORAGE	F	NEWFLG (6)	Code wants to generate more than NMXCLS interface cells. Redimension material arrays and redefine NMXCLS.
21.	JMAX = WHICH IS NOT AN EVEN NUMBER. THE GRID WAS NOT REZONED IN THE Y-DIRECTION	Nf	REZONE (156)	Grid must have an even number of rows to be rezoned in the Y-direction.
22.	MASS EVAPORATED DUE TO ROUND-OFF IN TPHASE I,J,MFLAG,AMX,T.E.	Nf	TPHASE (672)	Mass of a pure cell is too small. This should not occur frequently.
23.	MASS EVAPORATED DUE TO ROUND-OFF IN TPHASE, I,J,N,MFK,XMASS,SIE,RHO	Nf	TPHASE (700)	Mass of material N in an interface cell is too small or could not be evacuated. Not an error condition unless XMASS is relatively large.

	Error Message	Type	Source	Comment
24.	MIXED CELL MASS TRANSPORT ADJUSTED TO PREVENT OVER-EMPTYING I,J,M,N, SAMPY,SAMMP,SAMMY, SGAMC	Nf	DMADJ (59)	Not an error condition unless it recurs for the same cell on many successive cycles.
25.	NEW SLIP ENDPOINT NOT DEFINED L,SLOPS, SLOPT,TXCL,TYCL,TX1, TY1, TX2, TY2= ERROR IN PLGTCR	F	PLGTCR (118)	Plug surface extended into free surface cell but can't find an intercept with back of target. Check tracer particles of both packages.
26.	NEW SLIP ENDPOINT OUTSIDE GRID SLPNDX, SLPNDY,ALPHA,TXCL, TYCL. ERROR IN PLGTCR	F	PLGTCR (23)	Plug surface (slipline) can not be extended to a point outside the grid. Check the angle ALPHA.
27.	NUMBER OF PARTICLES HAS BEEN EXCEEDED	F	TSETUP (156)	Check input defining tracer particles as well as value assigned to NTPMX.
28.	NUMBER OF SLIDE CELLS EXCEEDS INPUT VALUE OF NSLD	F	PTSAV (69)	Redimension slide arrays and redefine NSLD.
29.	REZONE ROUTINE CAN NOT HANDLE PLUGGING PROBLEMS	F	REZONE (9)	Plugging calculations do not, in general, require rezoning.
30.	SUM OF FRACS NOT EQUAL TO TOTAL AREA I,J,TAU,XDY2PI =	Nf	FLGSET (191)	Interfaces may have crossed due to tracers becoming too sparse. If print occurs only once for a given cell, the error is caused by roundoff and can be ignored. Otherwise back up and move tracers manually in INPUT or have ADDTCR add tracers in that region. (See Section 7.4.) (This error can also occur if the interfaces are initially defined incorrectly.)

	Error Message	Type	Source	Comment
31.	TROUBLE DEFINING MATERIAL OF CELL THAT HAS BECOME PURE IN FLGSET, I,J =	F	FLGSET (145)	This can be a result of interfaces crossing. Look for error 31. It also can result from an isolated mass. The user can "evaporate" this mass manually in INPUT on a restart.
32.	TROUBLE WITH PRESSURE ITERATION I = J = ITC = PAV =	F	CDT (142 229)	This can be a result of too small a value of PMIN, or it can be caused by a discontinuous equation of state or one which has a negative slope in the P-V plane.

9.2 INTER DIAGNOSTIC PRINTS

Additional diagnostic prints can be generated by the user by appropriately defining the input variable, INTER. The table below indicates the nature of the diagnostic prints for each value (a prime number) of INTER. If the user wants more than one of the prints, set INTER equal to the product of the corresponding prime numbers. For example, if total energy is not being conserved, the user might set INTER = 5 x 7 to see which cell(s) are responsible for the error. In order to also see what the cell quantities are before and after each of the phases the user would set INTER = 3 x 5 x 7.

VALUE OF INTER	PRINTS FROM SUBROUTINE	NATURE OF INFORMATION PRINTED
2	CDT	All intermediate steps in pressure iteration for each multimaterial cell in active grid.
3	EDIT	An EDIT print of all cell quantities after both SPHASE and HPHASE, as well as the normal EDIT following CDT (on EDIT print cycles only).
5	TPHASE	Mass transport terms at each boundary of every cell, plus total energy in grid after each cell in active grid is updated in TPHASE.
7	SPHASE	Total energy in grid after each cell in active grid is updated in SPHASE.
11	UVMOD	Post-TPHASE material velocities in slip cells before and after the slipline conditions are invoked.
13	UVCALC	Post-HPHASE material velocities in slip cells before and after the slipline conditions are invoked.

NOTE: The user should be aware that invoking several of the above options and running a calculation for many cycles can generate a prodigious amount of output, particularly if there are many mixed cells and/or a large grid. Frequently, especially in the case of pressure iteration problems or an energy error, it is necessary to invoke the diagnostic prints for only one cycle in order to get the necessary information.

(The reverse of this page is blank)